£1.20

# BEEBUG

FOR THE BBC MICRO

Vol 3 No 10 APRIL 1985

**Mixed Modes**


Reviews
- AMX mouse
- Home-banking
- Logo
- Advanced Basic books

Features
- Mixing Modes
- Backwards
- Making music
- Brickie Nickie
- Disc formatting

And much more

**BRITAIN'S LARGEST COMPUTER USER GROUP**
**MEMBERSHIP EXCEEDS 30,000**

# EDITORIAL

## ACORN RESCUED BY OLIVETTI

As many members will be aware, Acorn were suffering major financial problems until rescued by Olivetti (Europe's second largest computer manufacturer) who now have a 49.3% stake in Acorn with an option to increase this to a controlling 50.1% in the future. The immediate consequences are a further round of redundancies bringing the total to approximately 120 (out of 450), the reorganization of Acorn into four new divisions, and the appointment of Dr Alex Reid as Chairman of Acorn Computers with Acorn's founders, Chris Curry and Herman Hauser, taking more of a back seat role.

We contacted Acorn to ask them some important questions.

**BEEBUG:** How do these changes affect Acorn's future support for the BBC micro?

**ACORN:** *The life of the BBC micro is already assured for at least another four years through the contract with the BBC, and the deal with Olivetti can only improve future support and development. The BBC micro will be the central product for the new Education and Training Division (which will also deal with production) and they will be looking, in particular, at expanding educational sales overseas.*

**BEEBUG:** What truth is there in the rumour that Acorn will effectively pull out of the home market once the BBC micro ceases to be profitable?

**ACORN:** *As already stated the BBC micro system has a continuing life of at least four years and there are no plans to ignore the home user market. The Consumer Division will be responsible for the Electron range and for marketing the BBC micro in the home user market. Development of new products will be actively pursued.*

**BEEBUG:** What will be the effect to the end user of the reorganization of Acorn into four divisions (Education & Training, Scientific & Industrial, Business, Consumer)?

**ACORN:** *Each division will be responsible for marketing within its particular area of interest. It will produce its own range of products and also market products from other divisions where appropriate. This reorganization will enable Acorn to focus more clearly and more strongly in each of these prime market areas.*

**BEEBUG:** How does this reorganization affect Acornsoft?

**ACORN:** *The various sections of Acornsoft will be much more closely related to the appropriate division within Acorn. "Acornsoft" will remain as the brand name for many of Acorn's software products.*

**BEEBUG:** What will be the effect on Customer Services?

**ACORN:** *Funtions such as Customer Services will in future be handled by each division. This will take a time to reorganize but technical queries about the BBC micro, for example, will be dealt with by Education and Training. It is also expected that dealers will become more specialised in the markets that they support, in line with Acorn's four marketing divisions, and become more heavily involved in supporting their customers.*

**BEEBUG:** What is likely to be the future for Acorn's ABC range?

**ACORN:** *This will be the responsibility of the new Scientific & Industrial Division. Marketing will benefit considerably from Olivetti's world wide experience in this field and there may be some changes in the range as a result.*

**BEEBUG:** What will happen with future products, such as Communicator and Video Disc?

**ACORN:** *New developments will continue as planned, though within the new divisional framework of Acorn. Thus interactive video will be within the Education and Training division, and this is already proving to be a very successful product.*

## BEEBUG'S FOURTH YEAR OF PUBLICATION

This issue marks the completion of Volume 3 of BEEBUG. We are already working on a host of new ideas for Volume 4, to offer even more value to members. To mark the first issue we shall, next month, be including a complete index to Volume 3, a voucher worth up to £3 against BEEBUGSOFT products and extra programs for the magazine cassette/disc, including a first class arcade game. Extra programs for this month's magazine/cassette are a colourful machine code game by Bob Anderson called Cosmonaut, a special program on artificial intelligence (?) J.M.O'Regan, and the full Spreadsheet program from the March and April issues.

# BEEBUG MAGAZINE

## GENERAL CONTENTS


2 Editorial
4 News
5 Of Mice and Micros
The AMX Mouse Reviewed
8 Backwards
9 Points Arising
10 Homelink for Homebanking
12 Castle Quest from Micro Power
13 Mixing Modes
19 Logo for the Beeb
23 A Spreadsheet Program (Part 2)
28 BEEBUG Workshop
Searching and Sorting (Part 1)
30 Understanding Disc Formatting
31 AMX Mouse Special Offer and Competition
32 Making Music on the Beeb (Part 3)
37 Scrabble Reviewed
38 Calculating the length of Programs
40 Basic in Depth
Two Books Reviewed
42 Beginners Start Here
Introducing Machine Code (Part 3)
44 Postbag
46 Brickie Nickie


**HINTS, TIPS & INFO**


12 Trouble with *FX138
18 Real Value of TOP
36 Bug in Assembler
37 OPENOUT Bug
39 Small OPENOUT Files
39 Printing HEX Numbers
39 OC Troubles in Wordwise
39 Zero Page Corruption
41 DATA Remarks
41 Recovering Lost Programs with the Z80
41 Z80 Basic String Bug
45 Correct Integrex Mode 7 Dumps
45 More Commands in Memoplan
45 Centering the Mode 7 Screen
45 Sidewise RAM Benefits


**PROGRAMS**


8 Backwards
13 Mixing Modes
23 Spreadsheet Program (Part 2)
28 Two Sorting Routines and Demo
32 Making Music Example
38 Calculating Program Length
42 Beginners Machine Code Examples
46 Brickie Nickie

# NEWS

TV 4 U

Channel 4 Television has started a new TV series that will interest most Beeb owners. "4 Computer Buffs" is screened on Mondays at 5.30pm. The programme will feature all the popular home micros with news and features to reflect the enthusiast's interests. There is a 'modem corner' for those interested in communications and special emphasis is being put on programming languages other than Basic. A lot of free software is being broadcast in various forms, all commissioned from professional software companies and of a high quality.

Some software is being broadcast via Channel 4's teletext service - 4-tel - but this is only for the Spectrum at present. There is also software broadcast as an audio tone to accompany the test card for half an hour on Tuesday mornings. During the programme a new method of software broadcasting is being tried. Using a light pen your computer can read software from a small section of the TV screen. Constructional details of the light pen are given in the programme itself.

THE PLOT THICKENS

If you're fed up with trying to fit yet another screen design onto the two graph paper pages provided at the back of the User Guide for that purpose, then Victory Educational has just the thing for you. The 'Screen plot' is a re-usable screen designer. With a graphics grid on one side and a text grid on the other, you write on it with special water-soluble (overhead projector type) pens and then wipe it all off again when you're through. Character designing grids are also provided along with a list of all the teletext graphics characters. Screen plot will cost you £11.99 including post and packing from Victory Educational on 0705-818635.

CHEAP(ISH) PLOTTERS

The Penman plotter is a novel plotter that breaks all the price barriers at £250. It is based on a turtle design, but is capable of high quality drawing. The three penned turtle moves around the paper sensing its position optically from the paper's edge. Driving software for the Beeb costs £25. A more conventional design comes from Linear Graphics. The Plotmate flatbed plotter costs £344 and will handle A4 paper and uses a single pen. Driving software that intercepts the Beeb's screen graphics commands and mirrors their actions on paper is included. Further details from Penman on 0903-209081 and Linear Graphics on 0702-541664.

ACORNSOFT GETS DOWN TO BUSINESS

Acornsoft has released a 'demonstration package' called 'Micros in Business' aimed at providing both an insight into what microcomputers can do for the office and also several usable pieces of software. The package is disc based and has sections on word processing, personnel, spreadsheet, and database. For the most part the programs are cut down versions of existing Acornsoft products. The whole package costs £59.80 from dealers.

NEW SOFTWARE

Proving that a programmer marches on his stomach, comes Comp-u-cater from Shumwari. This disc based menu planner costs £25. Monty Python's Terry Jones' book 'The Saga Of Erik The Viking' has now been blessed with an adventure game version. The game costs £9.95 from Mosaic and apparently is good enough to fool even Terry Jones. Equally steeped in fairy tale is 'Jack and the Beanstalk' from Superior Software, an arcade game at a price of £7.95. Also from Superior come 'Space Pilot' - beat up a variety of airborne adversaries - and 'Airlift' - return your compatriots to the safety of your helicopter base - both for £7.95. On a similar vein to the latter Pace offer 'Skyhawk' for £7.99 (£11.95 on disc) and also 'Sorcery' - an arcade adventure - for the same price.

# OF MICE AND MICROS

## The AMX Mouse Reviewed

**Can the AMX Mouse make an Apple out of the Beeb? Geoff Bains throws away his Beeb's keyboard and takes the package out for a test drive.**

| | |
|---|---|
| Product | : AMX mouse |
| Supplier | : Advanced Memory Systems, Green Lane, Appleton, Warrington, WA4 5NG. Ø925-62682 |
| Price | : £89.95 |

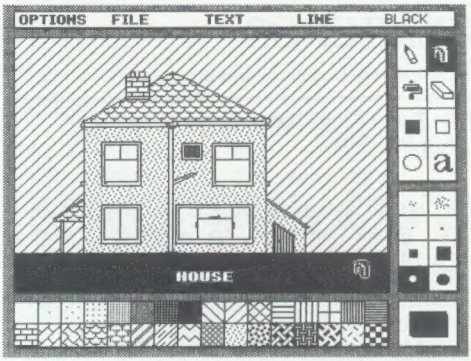
The idea of a friendly computer has taken a new turn in the past few years. Apple's Lisa and Macintosh computers along with the GEM package from Digital Research (to be available on Acorn's ABC) have brought with them a new buzz word - the mouse. A mouse is a device used on the table top next to the computer to control events on the computer's screen. Along with suitable software this little rodent claims to make computers usable by even the most computer illiterate amongst us. Now AMS have introduced the BBC micro to this exclusive club with the AMX mouse package.

The AMX mouse doesn't look very imposing. With the mouse comes a ROM, a cassette and disc, and two manuals. The mouse itself is a small black plastic box with a large metal ball-bearing mounted in its base. Rolling the mouse around on the desk top turns the ball bearing which communicates the mouse's movement to the computer via the user port. Three buttons in the front of the mouse can also be pressed to initiate actions much like the fire buttons on a joystick. The whole operation is managed by the AMX software sitting in the inevitable sideways ROM.

In many ways the mouse is no more than a new kind of joystick. You use it to guide a cursor around on the screen and press buttons to signify that you've got where you're going. However, without actually using one you cannot imagine how much superior to the joystick a mouse is. Within minutes of grabbing the critter you'll find it so natural to concentrate on the screen when using your computer and not have to worry about the keyboard - just as you would when using pen and paper.

Of course the mouse is only useful with the right applications software. For Basic programming or writing large amounts of text the keyboard is the only viable method to input your ideas to the computer - at the moment anyway. However, in some applications a mouse transforms a task from tedious in the extreme to sheer pleasure. The most obvious application for 'mouse technology' is in computer aided art and design. AMS has noticed this too and has included an excellent design program - AMX Art - in with the mouse package. AMX Art embodies all that is good about using mice.

The first thing to realise when using AMX Art is that there is only one reason to ever go near your keyboard when using this package. That is when entering file names. Everything else is controlled with the mouse and the screen. If you have taken up Apple's advertised offer of a Macintosh test drive or otherwise played with the Macdraw drawing package you will already know what AMX Art is all about. AMX art blatantly imitates that




package, and is all the better for it!

When you first boot up the software disc (or load the cassette - both are provided) a screen with several symbols, or 'Icons', presents itself. There is a symbol for the art program amongst them and moving the arrow shaped cursor, under mouse control, to the symbol and pressing one of the mouse buttons loads up AMX Art and sets it running. So far, so easy.

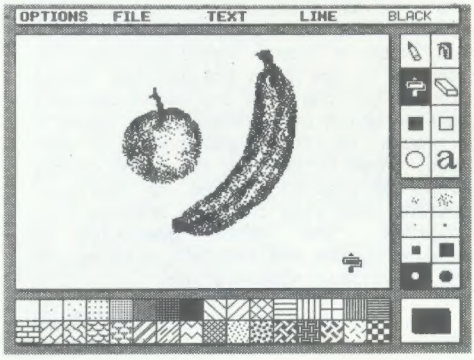
The AMX Art program itself is all on a single screen. All control is with the mouse with options selected from menus that you 'pull down' from the top with the cursor. Along the bottom of the screen is a selection of shading patterns and up one side a selection of drawing operations represented by symbols and another menu of line types.

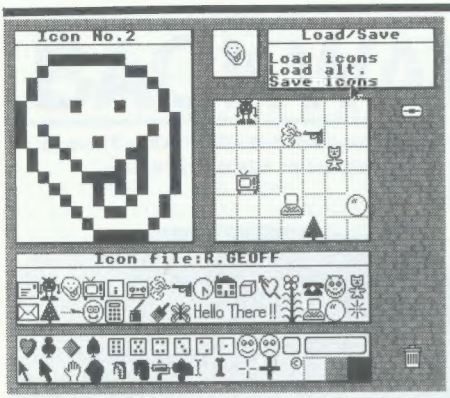
Drawing is done in the central region of the screen and is mastered within a matter of minutes. All you do is move the cursor with the mouse to symbol representing the drawing operation you want - pencil, eraser, spray gun, paint roller, etc. - and hit one of the mouse buttons. The cursor then changes shape to the symbol you've selected. Now select the line thickness you want from the other icon menu at the side of the screen and start drawing. You simply move the mouse around on the desk top and the cursor mirrors the movements on the screen - easy and natural. Pressing the mouse button will leave a line trailing behind the cursor. If you want to fill in an area then select the paint roller, select a shading pattern from the lower menu (the whole program deals only with black and white so differing patterns are used to fill in areas) and then move the paint roller cursor to the area and press the button. The area is quickly filled in, in your selected pattern.

The menus along the top of the screen are, apart from their titles, not normally visible. Moving the cursor to the title and pressing the button reveals the menu on top of your drawing. From these you can load and save pictures, select type style for text, change between cassette and disc, and so on. Once you've made your selection (with the mouse again), select the cancel option at the end of each menu and the menu disappears leaving your drawing unharmed underneath.

All this involves a good deal of moving back and forth across the screen between picture and menu, and no doubt sounds very lengthy and tedious. With any other control mechanism than a mouse, so it would be, but with the mouse the co-ordination between hand and eye, and mouse and screen, is so quick, easy, and natural that the whole process is almost as easy as drawing on paper. Except of course that you don't have problems with smudging, uneven and inaccurate shading, blunt pencils, and the like. Using AMX Art you can easily produce exact, impressive and even artistic pictures.

Needless to say, when you've finished your masterpiece there is an option (selected with the mouse of course) to dump the picture to a printer. Epson compatibles are dealt






with automatically but your own dump routine can be incorporated.

The program is not perfect. You can only draw in black and white on a mode 4 screen. The type styles are limited. The fill routine is not very sophisticated and needs several goes at a mildly complicated shape. Unlike Macdraw (on the Apple Macintosh) you cannot draw a small part of a picture and then move it around the screen or copy it to another place. However, these are comparisons with a package available on a machine with a 32 bit processor, 128K of RAM and costing many times the price of a Beeb. For a BBC micro program, AMX Art is not just good, but superb.

The Art program is not all you get in the mouse package. Also on the software disc is a character design program for creating your own icons. This is the easiest to use of any character designer I've ever seen. Again this is largely because of the mouse. These icons, or others already provided can be used in your own mouse controlled programs. In the AMX manuals there are detailed descriptions of how to incorporate the mouse control routines in the AMX ROM into your own programs. The mouse is polled, whenever it is moved, under interrupts and registers containing its position and the state of the buttons are all updated continuously. There are simple routines provided in the ROM to read these registers and make use of several other features such as moving the icon cursor around, creating windows, displaying an icon, adjusting the sensitivity of the mouse, and so on. These routines are accessed with *commands that you can incorporate into your own Basic program in much the same way as you might incorporate the commands available in Computer Concepts' graphics extension ROM or BEEBUGSOFT's sprite utilities package.

Okay, so the AMX mouse package gives you a fancy drawing package and a method of writing your own programs to make the most of mouse mania. But what about all the other tasks that your Beeb performs? Now you're hooked on mice how can you squeeze a rodent into them? Well unfortunately the answer is usually you can't. The mouse can be configured so that moving the mouse replaces the cursor keys, and the mouse buttons act instead of three keys of your choice so that you can use the mouse to a limited extent in other commercial programs. However, to really make use of the mouse a program has to be specially written with the mouse in mind. All actions must be menu based, preferably with extensive use of icons to save the user from having to read through vast quantities of text. As no other Beeb software is written in this way no amount of adaptability of the mouse really helps.

However, all is not lost. AMS do not plan to make the AMX mouse package a one off product but more the start of a new line of products. Already under way is a desk top manager (as used at the heart of the Lisa/Macintosh/GEM systems) giving you calculator, memo pad, diary, telephone directory, etc. all under mouse control, for about £25. A utility disc for the AMX Art package will provide a zoom facility and a teletext page designer for about £15. A colour Art program is also promised along with a database and even a utility to give you pull down menus in Wordwise. These are destined for summer release.

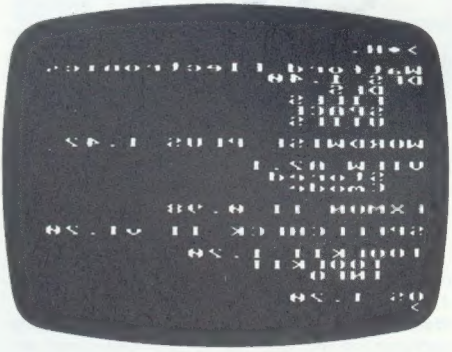
If these packages live up to the standards set by AMX Art then the whole AMX mouse system looks set to have a long and popular future. For the moment the AMX mouse package is worth buying for AMX Art alone. If you already use a drawing package of any description, if you intend to start using one, or even if you just fancy a fascinating new toy, then the present AMX mouse package has to be the best around.

# BACKWARDS TEXT

Tested on O.S. 1.2 Basic I & II

**If Lewis Caroll had ever written a computer program instead of 'Alice Through The Looking Glass' then maybe this is the program he might have written. Bill Wilkinson tells you what it's all about.**

Here is a handy little utility for fooling your friends. It's a routine to reverse the text displayed by your BBC. The program generates a short section of machine code which has this special effect on your BBC micro. After typing the program in, save a copy on disc or tape BEFORE running, because it will corrupt itself when run.



You can save the machine code section itself by running the program, pressing <Escape> when the "Press any key..." prompt is displayed, and typing:

```
*SAVE BACK A00 +FF
```

on a disc system, or

```
*SAVE BACK D01 +FF
```

on a tape system. The machine code can be re-loaded and executed by:

```
*RUN BACK
```

PROGRAM NOTES

The first part of the machine code 'explodes' the character set, redefines all the characters, and re-vectors the "write character" operating system routine (OSWRCH). It exits by executing a *BASIC, which has the effect of altering PAGE and HIMEM to take the enlarged character set and the change of screen mode into account. The second part is entered whenever a character is sent to the screen, and performs certain actions depending on the character code.

The text is mirrored in modes 0 to 6, i.e a greater than sign '>' becomes a less than sign '<' and the text begins from the right hand side of the screen instead of the left. In mode 7, because of the special teletext chip, the characters are not mirrored but again they start from the right hand side of the screen.

The value of PAGE is pushed up by the program because of the space needed to explode the character set for re-definition. The new value of PAGE will be PAGE+&600 bytes. So for cassette users, PAGE will be &1400 (&E00+&600), and most disc users will find PAGE at &1F00 (&1900+&600). The assembled machine code is located (by line 1020) at either &D01 for cassette systems or &A00 for disc systems. The program also resets the Break vector so that the effect can only be cancelled by switching off the machine!

```
  10 REM PROGRAM BACK
  20 REM VERSION B0.2
  30 REM AUTHOR  W G T Walker
  40 REM BEEBUG  APRIL 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 OSBYTE=&FFF4:OSWORD=&FFF1:OSASCI=
&FFE3
 110 OSWRCH=&FFEE:BLOCK=&70
 120 PROCassemble
 130 PRINT''''"Press any key when ready
...";:G=GET:*FX247 76
 140 A%=248:X%=res MOD 256:CALL&FFF4
 150 A%=249:X%=res DIV 256:CALL&FFF4
 160 CALL REDEF
 170 END
 180 :
1000 DEFPROCassemble
1010 FOR PASS=0 TO 3 STEP 3
1020 IF PAGE>&E00 THEN P%=&A00 ELSE P%
=&D01
1030 [OPT PASS
1040 \Initialisation
1050 \Explode character set
1060 .REDEF LDA #20:LDX #6:JSR OSBYTE ▶
```

```
 1070 \Redefine characters
 1080 LDA #33:STA BLOCK
 1090 \read char
 1100 .RLOOP1
 1110 LDA #10:LDX #(BLOCK AND &FF):LDY
#(BLOCK DIV &100):JSR OSWORD
 1120 \process char
 1130 LDX #8 ;for 8 bytes
 1140 .RLOOP2 LDY #8 ;for 8 bits
 1150 .RLOOP3 LSR BLOCK,X
 1160 ROL A
 1170 DEY:BNE RLOOP3
 1180 STA BLOCK,X
 1190 DEX:BNE RLOOP2
 1200 \redefine char
 1210 LDA #23:JSR OSWRCH
 1220 LDX #0
 1230 .DFBLK1 LDA BLOCK,X:JSR OSWRCH
 1240 INX:CPX #9:BMI DFBLK1
 1250 INC BLOCK:BPL RLOOP1
 1260 \re-vector OSWRCH
 1270 .res:LDA &20E:STA IO+1
 1280 LDA &20F:STA IO+2
 1290 LDA #(INV MOD &100):STA &20E
 1300 LDA #(INV DIV &100):STA &20F
 1310 \select MODE 6
 1320 LDA #22:JSR IO:LDA #6:JSR IO
 1330 LDA #12:JSR INV
 1340 \execute *BASIC to rewrite PAGE a
nd HIMEM
 1350 LDX #(STR MOD &100)
 1360 LDY #(STR DIV &100)
 1370 JSR &FFF7
 1380 \alternative OSWRCH
 1390 .INV
 1400 STA ASTORE:TXA:PHA:TYA:PHA
 1410 \read VDU queue length
 1420 LDA #&DA:LDX #0:LDY #255
 1430 JSR OSBYTE
 1440 LDA ASTORE
 1450 CPX #0:BEQ INV1
 1460 \print char and quit
 1470 .IOT JSR IO
 1480 .REST PLA:TAY:PLA:TAX:LDA ASTORE
 1490 RTS
 1500 \delete?
 1510 .INV1 CMP #127:BNE INV2
 1520 LDA #9:JSR IO:LDA #32:JSR IO
 1530 \backspace and quit
 1540 .BACKOUT LDA #8:BNE IOT
 1550 \control code?
 1560 .INV2 CMP #32:BCS CHOUT
 1570 \tab?
 1580 CMP #9:BEQ BACKOUT
 1590 \backspace?
 1600 CMP #8:BNE INV3
 1610 LDA #9:BNE IOT
 1620 \cls?
 1630 .INV3 CMP #12:BNE INV4
 1640 JSR IO:LDA #30
 1650 \home?
 1660 .INV4 CMP #30:BNE INV5
 1670 JSR IO:LDA #10:JSR IO:JMP BACKOUT
 1680 \carriage return?
 1690 .INV5 CMP #13:BNE IOT
 1700 LDA #10:JSR IO:LDA #13:JSR IO
 1710 JMP BACKOUT
 1720 \print non-control-char
 1730 .CHOUT JSR IO:LDA #8:JSR IO
 1740 \read cursor position
 1750 LDA #134:JSR OSBYTE
 1760 TXA:BNE BACKOUT
 1770 LDA #10:JSR IO:JSR IO
 1780 JMP BACKOUT
 1790 .IO JMP &FFFE
 1800 ]
 1810 ASTORE=P%:P%=P%+1
 1820 STR=P%:$P%="BASIC"+CHR$(13)
 1830 NEXT
 1840 ENDPROC
```

## POINTS ARISING

REVIEW OF ROM EXPANSION BOARDS (BEEBUG Vol.3 No.6)

Regrettably a small error of fact arose in compiling the table at the start of this review. Because of the way the Aries B-12 board is installed, the maximum number of ROMs that can be accommodated is 12 and not 16 (column CIT). There is also an additional charge of £5.75 for the adaptor board, required if the Aries B-20 memory expansion board is not already fitted.

REVIEW OF PHLOOPY (BEEBUG Vol.3 No.7)

Phi Mag Systems Ltd have advised us that the price of the Phloopy has been reduced to £117.85 including VAT and post & packing.

REVIEW OF SQUASH (in Adventure Games, BEEBUG Vol.3 No.8)

Prices for this program have been reduced to £7.95 (cassette), £9.75 (5" disc), and £11.75 (3" disc).

# HOMELINK FOR HOMEBANKING

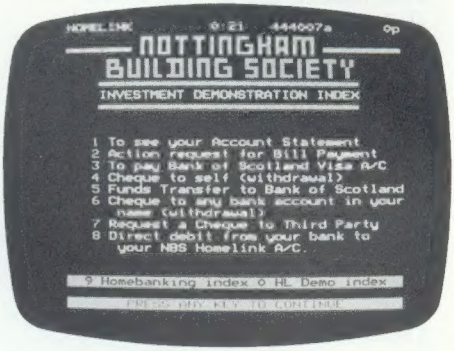
**The Midland Homebanking system was described in a previous issue of BEEBUG (Vol. 3 No. 6). David Turner now describes an alternative service being offered jointly by the Nottingham Building Society and the Bank of Scotland.**

These notes about the Homelink on-screen banking service are not presented as a dispassionate review but are written from the viewpoint of an enthusiastic user. However, the author has no connection with Homelink apart from being a user, so there is no commercial motive.


Homelink is not an experiment. It is a fully developed system run jointly by the Nottingham Building Society and the Bank of Scotland. It is not complicated to use. Once Prestel has been joined (existing Prestel users obviously benefit here) running costs are small. There are no account charges if account balances are maintained above a reasonable level and because it is open at night, work can be done at times when local phone calls cost only 40p per hour and Prestel computer time is free. It enables members to do all the following (and more) from their computer keyboards:

1. Call up on screen their Bank of Scotland or Nottingham Building Society account statements, with interest shown up to the current day.

2. Transfer funds, either instantly or at specified future dates, from the Building Society to the Bank of Scotland (transfers in the other direction occur overnight).



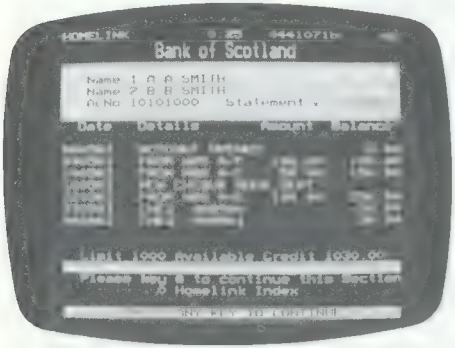

3. Pay registered bills, credit card accounts, etc.

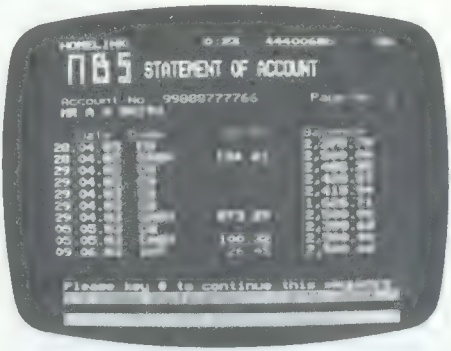
4. Buy and sell shares on the stock market.

5. Apply for mortgages and loans, or see a current mortgage statement if they already have one.

6. Send messages (including confidential ones) to both building society and bank and receive replies on-screen.

It goes without saying that security is of paramount importance in any banking service. Homelink is particularly proud of its system. First the normal Prestel security system must be negotiated. Then, before any transaction involving real money is allowed, three separate security codes must be typed in. They do not appear on the screen as they are typed, but are replaced by dashes to prevent other people reading them. The first is the user's account number. This hardly counts as security as anyone with access to the statements and pass-books could read it. The second item, the Personal Identification Number (PIN), is totally under the user's control. It is not a mere four digit PIN like those issued by banks for use with cash

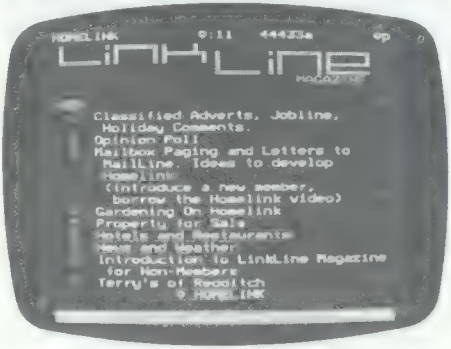



dispenser cards, but a code comprising any mixture of four to ten digits or letters chosen by the user and then registered with the Homelink computer. It can be changed as often as you like at any time of the night or day. The third item is a Transaction Number. This is required to be the next from a sequence of four-digit random numbers allocated to the user by the Homelink computer. When needed, the computer sends another batch by mail. Each number is used once only. A successfully completed transaction is confirmed by a message frame which reminds the user to delete the used number from the list.

It is difficult to think of any way this system might be beaten, so long as the PIN is not written down, but as an added precaution the account is frozen if three incorrect attempts are made to access it. Written authority is then needed to reactivate it. The PIN selected therefore needs to be memorable, but not predictable. One's first wife's maiden name would not be a good choice!

It is not necessary to own a BBC Micro to receive the benefits of Homelink on-screen banking, but Prestel membership is essential and is normally applied for at the same time as Homelink. The system can be used with lesser micros, or even without a microcomputer at all with a 'Home Deck' loaned or bought from NBS. However BBC Micro owners are already halfway there. Those already using Prestel and with modest savings that could be switched to NBS can join for nothing. I find the system of great practical value, justifying the cost of a Prestel subscription on its own.




The two organizations running it are very friendly and helpful, and answer messages promptly. The interest rates compare well with the best on offer, yet money invested is instantly available. The Bank of Scotland account operates as an ordinary current account, with a normal cheque book, the one difference being that interest is added monthly to credit balances. A Visa card is available that doubles as a £50 cheque guarantee card and can be used to obtain cash. It is not even necessary to change existing banking arrangements, as the Band of Scotland account may be used in parallel.

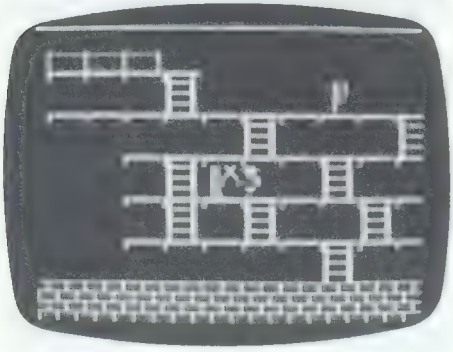
Information on Homelink can be obtained from:-

Nottingham Building Society (B),
5/13 Upper Parliament Street,
Nottingham,
NG1 2BX.

# CASTLE QUEST

**Micropower are claiming that their latest release, Castle Quest, is probably the most challenging game ever released for the BBC micro. Our resident games enthusiast, Alan Webster, has been following the quest for treasure. Is this the game we'll all be playing in 1985?**

```
Title    : Castle Quest
Supplier : Micro Power
Price    : Cassette £12.95
           Disc £14.95
Rating   : *****
```



'Manic Miner' type games are all the rage at the moment. These are arcade-style action games where you have to tackle a set of problems, much as in an adventure, to reach a final goal.

'Castle Quest' is probably the most difficult of these games so far for the Beeb, and definitely the most attractive as far as action and graphics are concerned. It is claimed to be 'Probably the most challenging game ever devised for the BBC Micro', and after a few hours of play I realised that I was not going to get very far in a hurry.

The object of the game is to find the wizard's treasure, hidden in a castle full of troll's, wicked witches and spiders. One of the first challenges you are likely to encounter is how to escape from a guarded dungeon armed, apparently, with only a stool. Guile and deception are the answers here, rather than any unsubtle fighting.

The game involves negotiating ladders and walkways and leaping across voids in true arcade style, whilst at the same time trying to solve the puzzles and collect items to help you in your quest to find the treasure.

'Castle Quest' uses thirteen different keys (a measure of the complexity of the game?) and also features so called 'MP4 Scrollerama', Micro Power's super-smooth 4-direction scroll.

The packaging is above Micro Power's usual standard, but even the disc version did not contain on-screen instructions.

Micro Power are hoping that this will be the first of many successful action/adventure/puzzle type games, but the most challenging game for the Beeb? In my opinion the answer is yes, but some would say that Acornsoft's Elite was a more challenging game, it all depends on the sort of game you like.

As a final incentive to buy the game, Micro Power are betting you £1 that you cannot finish the game within three month's of purchase. If you do crack the puzzle within this time, you not only get your £1, but free entry into the £500 challenge. For more information on 'Castle Quest' see your local dealer. It's well worth it.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

TROUBLE WITH *FX138 - P.D. Mercer

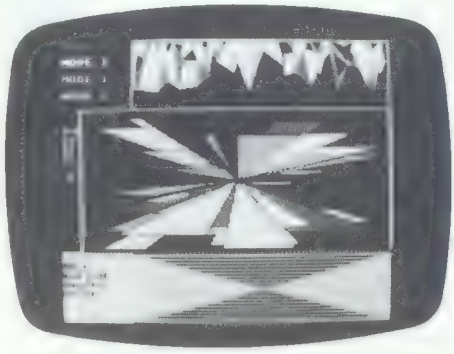
Using *FX138 repeatedly to insert more than 31 characters into the keyboard buffer results in only the first 31 being accepted.

Tested on O.S. 1.2
Basic I & II

# MIXING MODES

**If you thought that it was impossible to have mode 0, mode 1 and mode 2 displays on the screen at the same time then think again. Ian Hall has come up with an ingenious program that will allow your programs to mix modes with obvious benefits. Sixteen colours and 80 column text? It's all possible now.**

This program will mix different display modes on the screen simultaneously. Up until now you've only been able to do that playing Elite! Of course BEEBUG goes one better than that by giving you THREE modes at once. You can mix different modes to give, say, 80 column text along with 16 colour graphics, or mix different coloured versions of the same mode to give, say, a 12 colour mode 1. The routines can be easily incorporated into your own programs giving you little short of a new computer.



## THE DEMONSTRATION

The programs are in the form of a demonstration, parts of which can easily be adapted and incorporated into your own programs for your own uses.

To run the demonstration type in both listings and save each before running it. Save the second program with the name 'MIXDEMO' as this is chained from the first using that name.

The demonstration divides the screen horizontally into three areas which have moving mode 1, 2, and 0 displays, respectively, in them.

It is quite easy to use parts of these programs in your own Basic programs to create your own mixed mode displays. You will need to use all of the first (MIXMODE) program, with some alterations, and a short section of the MIXDEMO program.

The MIXMODE program sets up the machine code that looks after the display of the three modes simultaneously, and then chains to the second program, whether your own or the demo given here, to create the displays.

## SETTING UP MIXED MODE SCREENS

One section of the MIXMODE program must be altered to set up the particular mixture of modes you wish to display. This is the procedure PROCsetmodes.

The three mode sections displayed are designated (from top to bottom) section A, section B, and section C. You should first plan what modes you wish to appear on the screen and the size and position of each section. The mode for each section should be entered into PROCsetmodes as VDU22 statements in lines 1130, 1170, and 1210.

You can also enter information such as the graphics and text windows required for each section of the screen (as done in the listing given) or VDU19 colour definitions here but this is not essential. More on that later.

At the end of each section definition (lines 1160, 1200, and 1240) the information is stored away with a call to the relevant 'push' routine for that section.

Finally the two parameters that decide where the screen is to be split must be set up. These are held in the locations delay1 and delay2 and must be set up in line 1250. The following formulae are used to determine the approximate values to be placed in

delay1 and delay2:

```
!delay1=1725+16300*(ha/1024)
!delay2=16300*((ha+hb)/1024)
```

where ha and hb are the heights, in graphics co-ordinates, of the two top sections, A and B, respectively in the mixed mode screen.

Some slight tweeking of these values may be necessary to get the mode split in exactly the right position since there is slight jitter on one graphics line at the join between two modes. If you only want a two mode display then set !delay2 to &FFFF and do not use pushc or pullc in the programs. If you want to create a screen using the 10K modes (4 and 5) rather than using any of the 20K modes (0, 1, and 2) then change line 100 of the MIXMODE program to MODE 4 and the PROCsetmodes lines to use VDU22,4 and VDU22,5 as needed.

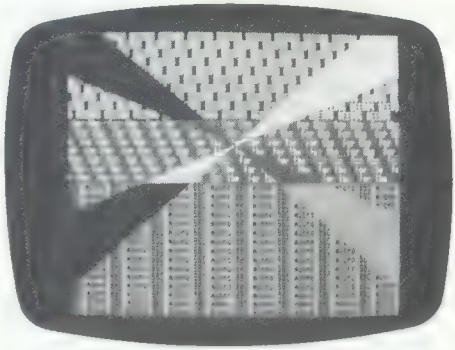
This completes the customisation of the MIXMODE program. This should now be saved.

USING THE MIXED MODE SCREEN

Once each mode area has been defined by running the first program, your own program to use the mixed mode display must include PROCinit from the MIXDEMO program and a call to that procedure (line 110 in the MIXDEMO program). The mixed mode display is enabled with 'CALL enable' (this is done in the MIXDEMO program also in line 110). Once that has also been called, it is as though you have three totally separate smaller displays operating on the same screen at once. You call each one up with a CALL to the relevant 'pull' routine. For example, to print something in the section A area and then draw a triangle in the section B area, the following is performed:

```
CALL pulla
PRINT "Hello"
CALL pullb
MOVEx1,y1:MOVEx2,y2:PLOT85,x3,y3
```

If you change any of the section information, such as window sizes, current graphics and text cursor positions, GCOL information etc. when



using any section, this can be stored as relevant to that section alone with a CALL to the relevant 'push' routine. When you next 'pull' that section for use that information will all be re-established. So in the next example the triangle in section A is completed correctly despite the intervening section B action because the graphics cursor position is stored away by the 'CALL pusha'.

```
CALL pulla
MOVE 100,900:MOVE 200,900
CALL pusha
CALL pullb
MOVE 500,500:PRINT "HELLO"
CALL pulla
PLOT 85,150,1000
```

If you are not changing the section information (VDU19 colours, windows, and so on) as you use the mix mode screen, then the pushing of this information can be done when setting up the modes in PROCsetmodes in the MIXMODE program, to establish start up conditions for each section. If your screen comprises different modes it is recommended that you define text and graphics windows for each section of the display and 'push' this information in the MIXMODE program. A whole screen window in a mixed mode display will operate as normal but text and graphics that are printed from within one mode will not be displayed correctly in the others.

However a slightly different application of this program is to create a screen of three versions of the same mode, each with different

VDU19 colour definitions (giving, say, a 12 colour mode 1 screen). If you make use of this facility then you can leave the windows at their default value of the entire screen. Text and graphics can now be placed on the screen as normal but will appear in the different colours in the different sections.

Initially you may find it useful to experiment with the demonstration to explore the full potential of these programs.

LOCATING THE CODE

The machine code that controls the mixed mode display occupies four pages (&400) of memory and can be placed where you want it. This is achieved by using the desired start address as the passed parameter when calling the PROClocate() procedure in the MIXMODE program (line 120). In the demonstration the code is placed just below screen memory. HIMEM is adjusted so that the code fits immediately above it (lines 110 and 120). This is suitable for most Basic applications.

VDU 19

The use of the VDU19 command will not change logical colour until the appropriate mode is 'pushed' away. You can experiment with this by using Escape to exit from the demonstration and play with calling the push and pull routines in immediate mode.

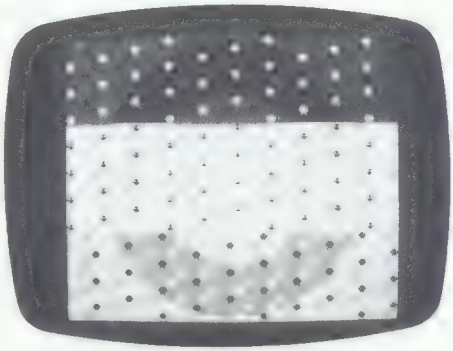
The use of a VDU19 command may produce a glitch on the screen but this can be avoided by using the following command in place of VDU19,n,m;0; (see line 1460 of the MIXDEMO program):

```
colour?n=m
```

As with the VDU19 command the change of logical colour will not take place until the appropriate push routine is called.

LIMITATIONS

One limitation of this program is that, to avoid screen jitter, the system interrupts have to be carefully controlled. To this end, the analogue input has been disabled (line 1600 in the first program) since this constantly interrupts the processor. The ADC inputs can however be read (to



8 bit accuracy) with the following function. Converting this to a machine code routine will speed up the conversion if required:

```
DEFFNadc(chan%)
?&FEC0=(chan%-1)AND3
REPEAT UNTIL (?&FEC0 AND &80)=0
=?&FEC1
```

This function will directly access the ADC without the need to interrupt the processor.

As considerable access is made to the operating system workspace and to some SHEILA hardware addresses, for reasons of speed, operating system calls are not used. Thus this program will not operate over the Tube.

Next month Ian Hall describe some of the ideas and techniques used in these programs.

```
 10 REM PROGRAM MIXMODE
 20 REM VERSION B0.1
 30 REM AUTHOR  Ian Hall
 40 REM BEEBUG  APRIL 1985
 50 REM PROGRAM SUBJECT TO COPYRIGHT
 60 :
 70 ON ERROR GOTO 3490
100 MODE 0
110 HIMEM=HIMEM-&400
120 PROClocate(HIMEM)
130 PROCassemble
140 PROCsetmodes
150 CHAIN"MIXDEMO"
160 END
```

```
  170 DATA0,0,0,0,0,0,0,0,1,1,1,1,1,1,1
,1,0,0,1,1,0,0,1,1,2,2,3,3,2,2,3,3,0,1,
2,3,4,5,6,7,8,9,10,11,12,13,14,15
  180 :
 1000 DEFPROCclocate(a%)
 1010 code%=a%  :a%=a%+480
 1020 vduvara=a%:a%=a%+&A0
 1030 vduvarb=a%:a%=a%+&A0
 1040 vduvarc=a%:a%=a%+&A0
 1050 palette=a%:a%=a%+48
 1060 delay1=a% :a%=a%+4
 1070 delay2=a%
 1080 FOR I%=0TO47:READ palette?I%:NEXT
 1090 ENDPROC
 1100 :
 1110 DEFPROCsetmodes
 1120 *TV0,1
 1130 VDU22,1:REM MODE 1
 1140 VDU24,279;768;1279;1023;
 1150 VDU28,0,7,39,0
 1160 CALL pusha
 1170 VDU22,2:REM MODE 2
 1180 VDU24,79;256;1279;767;
 1190 VDU28,0,23,19,8
 1200 CALL pushb
 1210 VDU22,0:REM MODE 0
 1220 VDU24,279;0;1279;255;
 1230 VDU28,0,31,79,24
 1240 CALL pushc
 1250 !delay1=5800:!delay2=8150
 1260 ENDPROC
 1270 :
 1280 DEFPROCassemble
 1290 osbyte=&FFF4
 1300 colour=&36F:H%=colour
 1310 oldint=&70:oldeve=&72
 1320 vdu=&74
 1330 temp1=&76:temp2=&77
 1340 mask1=&78:mask2=&79
 1350 state=&7A
 1360 ?mask1=&40:?mask2=&02
 1370 :
 1380 FOR I%=0 TO 2 STEP 2
 1390 P%=code%
 1400 [OPTI%
 1410 :
 1420 .enable:.A%
 1430 LDA &204:STA oldint
 1440 LDA &205:STA oldint+1
 1450 LDA &220:STA oldeve
 1460 LDA &221:STA oldeve+1
 1470 LDA #event AND &FF
 1480 STA &220
 1490 LDA #event DIV &100
 1500 STA &221
 1510 SEI
 1520 LDA #inter AND &FF
 1530 STA &204
 1540 LDA #inter DIV &100
 1550 STA &205
 1560 CLI
 1570 LDA #14:LDX #4
 1580 JSR osbyte
 1590 LDA #189:LDX #0:LDY #0
 1600 JSR osbyte
 1610 LDA &FE6B
 1620 AND #&3F
 1630 EOR #&40
 1640 STA &FE6B
 1650 RTS
 1660 :
 1670 .event
 1680 PHP
 1690 SEI
 1700 PHA:TXA:PHA
 1710 LDA #0:STA state
 1720 LDA #1000 AND &FF
 1730 STA &FE44
 1740 LDA #1000 DIV &100
 1750 STA &FE45
 1760 LDA #10000 AND &FF
 1770 STA &FE46
 1780 LDA #10000 DIV &100
 1790 STA &FE47
 1800 ROR vduvara+&10
 1810 LDA &248:ROR A
 1820 ROL vduvara+&10
 1830 ROR vduvarb+&10
 1840 LDA &248:ROR A
 1850 ROL vduvarb+&10
 1860 ROR vduvarc+&10
 1870 LDA &248:ROR A
 1880 ROL vduvarc+&10
 1890 JSR screena
 1900 PLA:TAX:PLA:PLP
 1910 JMP (oldeve)
 1920 :
 1930 .inter
 1940 PHP
 1950 SEI
 1960 PHA
 1970 LDA &FE4D:BPL hop
 1980 BIT mask2:BNE sync
 1990 .hop
 2000 LDA &FE6D:BPL ret
 2010 BIT mask1:BNE change
 2020 .ret
 2030 PLA:PLP
 2040 JMP (oldint)
 2050 :
 2060 .sync
 2070 LDA delay1  :STA &FE64
 2080 LDA delay1+1:STA &FE65
 2090 LDA #&C0    :STA &FE6E
 2100 LDA delay2  :STA &FE66
 2110 LDA delay2+1:STA &FE67
 2120 JMP ret
 2130 :
 2140 .change
 2150 STA &FE6D
```

```
2160 TXA:PHA
2170 LDA state
2180 CMP #0:BEQ state0
2190 CMP #1:BEQ state1
2200 .return
2210 PLA:TAX:PLA:PLP
2220 JMP (oldint)
2230 :
2240 .state0
2250 JSR screenb
2260 INC state
2270 JMP return
2280 .state1
2290 JSR screenc
2300 INC state
2310 LDA #&40:STA &FE6E
2320 JMP return
2330 :
2340 .screena
2350 LDA vduvara+&10:STA &FE20
2360 LDX #15
2370 .lo3
2380 LDA vduvara,X:STA &FE21
2390 DEX:BPL lo3
2400 RTS
2410 .screenb
2420 LDA vduvarb+&10:STA &FE20
2430 LDX #15
2440 .lo2
2450 LDA vduvarb,X:STA &FE21
2460 DEX:BPL lo2
2470 RTS
2480 .screenc
2490 LDX #15
2500 .lo1
2510 LDA vduvarc,X:STA &FE21
2520 DEX:BPL lo1
2530 LDA vduvarc+&10:STA &FE20
2540 RTS
2550 :
2560 .seta
2570 LDA #vduvara AND &FF
2580 STA vdu
2590 LDA #vduvara DIV &100
2600 STA vdu+1
2610 RTS
2620 .setb
2630 LDA #vduvarb AND &FF
2640 STA vdu
2650 LDA #vduvarb DIV &100
2660 STA vdu+1
2670 RTS
2680 .setc
2690 LDA #vduvarc AND &FF
2700 STA vdu
2710 LDA #vduvarc DIV &100
2720 STA vdu+1
2730 RTS
2740 :
2750 .pusha:.B%
2760 JSR seta
2770 JMP push
2780 .pushb:.C%
2790 JSR setb
2800 JMP push
2810 .pushc:.D%
2820 JSR setc
2830 .push
2840 LDY #&10
2850 LDA &248:STA (vdu),Y
2860 INY
2870 LDA &249:STA (vdu),Y
2880 INY
2890 LDX #0
2900 .loop1
2910 LDA &D0,X:STA (vdu),Y
2920 INY:INX
2930 CPX #10:BNE loop1
2940 LDY #&20:LDX #&0
2950 .loop2
2960 LDA &300,X:STA (vdu),Y
2970 INY:INX
2980 CPX #&80:BNE loop2
2990 LDY #&75
3000 LDA (vdu),Y:AND #&03
3010 ASL A:ASL A:ASL A:ASL A
3020 STA temp1
3030 LDX #0
3040 .loop3
3050 LDY temp1
3060 LDA palette,Y
3070 TAY
3080 LDA colour,Y:STA temp2
3090 TXA:TAY
3100 ASL A:ASL A:ASL A:ASL A
3110 CLC
3120 ADC temp2
3130 EOR #&07
3140 STA (vdu),Y
3150 INX:INC temp1
3160 CPX #&10:BNE loop3
3170 RTS
3180 :
3190 .pulla:.E%
3200 JSR seta
3210 JMP pull
3220 .pullb:.F%
3230 JSR setb
3240 JMP pull
3250 .pullc:.G%
3260 JSR setc
3270 .pull
3280 LDY #&10
3290 LDA (vdu),Y:STA &248
3300 INY
3310 LDA (vdu),Y:STA &249
3320 INY
3330 LDX #0
3340 .loop4
3350 LDA (vdu),Y:STA &D0,X
```

```
3360 INY:INX
3370 CPX #10:BNE loop4
3380 LDY #&20
3390 LDX #0
3400 .loop5
3410 LDA (vdu),Y:STA &300,X
3420 INY:INX
3430 CPX #&80:BNE loop5
3440 RTS
3450 ]
3460 NEXT
3470 ENDPROC
3480 :
3490 ON ERROR OFF
3500 MODE 7
3510 IF ERR<>17 THEN REPORT:PRINT " at
line ";ERL
3520 END
```

```
10 REM PROGRAM MIXMODE DEMO
20 REM VERSION B0.1
30 REM AUTHOR  Ian Hall
40 REM BEEBUG  APRIL 1985
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
100 ON ERROR GOTO 1530
110 PROCinit:CALL enable
120 PROCsetscreen
130 PROCdraw
140 END
150 :
1000 DEFPROCinit
1010 enable=A%
1020 pusha=B%:pushb=C%:pushc=D%
1030 pulla=E%:pullb=F%:pullc=G%
1040 colour=H%
1050 ENDPROC
1060 :
1070 DEFPROCsetscreen
1080 CALL pulla
1090 CLS
1100 VDU29,279;768;
1110 VDU19,2,2;0;
1120 VDU23,1,0;0;0;0;
1130 PRINTTAB(0,7)
1140 CALL pusha
1150 CALL pullb
1160 CLS
1170 VDU29,79;256;
1180 VDU23,1,0;0;0;0;
1190 CALL pushb
1200 CALL pullc
1210 CLS
1220 VDU29,779;128;
1230 VDU19,1,0;0;
1240 VDU23,1,0;0;0;0;
1250 CALL pushc
1260 ENDPROC
1270 :
1280 DEFPROCdraw
1290 C%=1:X%=1200:Y%=0:I%=200:K%=1000
1300 REPEAT
1310 CALL pulla
1320 Y%=RND(255):X%=X%+RND(50)
1330 IF X%>1000 THEN X%=0:COLOUR(C%):P
RINT'"MODE 1":GCOL0,3:MOVE1000,0:DRAW10
00,255:DRAW0,255:DRAW0,0
1340 C%=(C%+1)MOD3+1:GCOL0,C%
1350 PLOT 85,X%,Y%
1360 CALL pusha
1370 CALL pullb
1380 K%=K%+1:IF K%>70 THEN K%=0:GCOL0,
7:CLS:MOVE0,0:DRAW 1200,0:DRAW 1200,511
:DRAW 0,511:DRAW 0,0:COLOUR(RND(3)+8):P
RINTTAB(0,2)"M"'"O"'"D"'"E"'"2"
1390 GCOL0,RND(8)-1
1400 W%=RND(350)+50:T%=RND(1200-W%):S%
=RND(495)+8
1410 MOVE 600,256:MOVE T%,S%:PLOT 85,T
%+W%,S%
1420 GCOL0,0
1430 DRAW T%,S%:DRAW 600,256:DRAW T%+W
%,S%
1440 CALL pushb
1450 CALL pullc
1460 I%=I%+6:IF I%>=120 THEN I%=0:GCOL
0,1:MOVE-500,-128:DRAW 500,-128:DRAW 50
0,127:DRAW -500,127:DRAW -500,-128:PRIN
TTAB(0,1)"MODE 0"'"Which is 80"'"charac
ters"'"per line":A%=RND(7):colour?0=A%
1470 J%=I%*4
1480 GCOL3,1:MOVE-J%,-I%:DRAW-J%,I%:DR
AWJ%,I%:DRAWJ%,-I%:DRAW-J%,-I%
1490 CALL pushc
1500 UNTIL FALSE
1510 ENDPROC
1520 :
1530 ON ERROR OFF
1540 IF ERR<>17 THEN REPORT:PRINT " at
line ";ERL
1550 END
```

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

REAL VALUE OF TOP - T.K. Cowell

Although the value of TOP will tell you where your program ends in memory it won't tell you the limits of the space required for variables when the program is run. For this, run the program once and type:

```
DIM T 1:PRINT T
```

This will return the first free byte above program AND variable storage.

# LOGO FOR THE BEEB

**Logo is a very different kind of computer language that has for long provoked interest in the educational world. So far only limited 'Turtle Graphics' versions have been available for the Beeb, but now everything is changing with a full implementation of this language. And not just one, but four different versions have all been launched very recently. Mark Sealey, a teacher and enthusiast of Logo, has been finding what all the fuss is about.**

| | |
|---|---|
| Acornsoft (2 ROMS) | £69.00 |
| BBC Soft/Open University (2 ROMS) | £69.95 |
| Logotron/LCSI (16K ROM) | £67.78 |
| Logo Software Ltd. (16K EPROM) | £67.34 |

Prices included VAT.

INTRODUCTION

Logo was developed from Lisp by the American Seymour Papert and others in the late 1960s specifically for educational use; until now it has mainly been used in this country in schools to allow younger children to explore mathematical ideas through programming. The BBC micro has "suffered" from a variety of half-versions which excluded all but crude turtle graphics (hence its popular association with graphics). In fact Logo deals in a sophisticated way with text as well as arithmetic, logic and data functions.

Logo, like Pascal, does not use line numbers. Instead a sequence of commands is built up and tested line-by-line in 'immediate mode', as it were. Once you are happy with them, they may be stored in memory as a 'Procedure', designed to do a short, self-contained task. Procedures are then linked, called or nested in the order in which they will be needed in a longer program. Subsequently they can be amended or improved with an 'editor'. The equivalents to Basic's keywords are 'Primitives', which usually require an 'input', thus:

```
To Triangle
        REPEAT 3
        FD 50          (forward 50)
        LT 120         (left turn 120)
```

'Triangle' has now joined Logo's repertoire and can be called at will and as often as needed.

Four versions of the Logo language have recently appeared. As detailed differences between implementations are highlighted in the tables, emphasis has been given to the extent to which children of all ages will find the versions easy and reliable to use. After all, these ROMs are largely aimed at the educational market. High priority has also been given to overall user friendliness.

Acornsoft and BBC/OU Logo come with Extension/Applications discs (available from the other two suppliers shortly). The example programs on the Acornsoft one are particularly impressive.

THE LOGO ENVIRONMENT

Papert conceived of a 'Microworld', a truly functioning environment in which the user can think THROUGH the computer very often using words, rather than abstruse formulae (as in Basic, say). His view is of computers and computer languages as "tools to think with". As in all successful learning you explore in small steps from the familiar to the new. The Logotron version was most clearly conceived with this philosophy to the fore.

Various features enhance this aspect of Logo. Sprites, for instance, are completely programmable coloured screen objects capable of moving freely (arcade-like) as fast as the single, simple arrowhead 'turtle' used to mark the drawing-nib in Logo graphics.

Logotron's plans for a dedicated sprite board are the most advanced. (This should be now be available at around £130 + VAT). Their version also has a USE command for future interface options (floor turtles, sprites, robots etc.)...a healthy policy of development rather than producing a once and only 'product'; they are also supporting software written to illustrate (literally) Logo programs for children in the "Tilley the Turtle" animated story series. LSL also have a similar board under development.

At the same time, Acornsoft's version features an EXPLORE command that returns the distance a floor turtle has travelled before hitting a physical object and the facility for procedures to define (not CALL) other procedures at run-time. The BBC/OU version has a DRIVE primitive to move either floor or screen turtle in real-time i.e. synchronous with a key being held down.

Many of the shortcomings (e.g. no PI) in the two one-chip versions can often be got around (usually by an OS command), others less easily so (no equivalent to Basic's GET in LSL's ). A comparison of the more important features can be found in Table 1.

All four versions provide complete access to the BBC OS commands, VDU and Envelope/Sound commands. All will work with both tape and disc and claim 6502 second processor compatibility (BBC/OU with disc patch). The turtles are or will be driven by disc extensions.

GRAPHICS

Acornsoft Logo has the most solid feel to its screen and has extra help in that the border changes according to FENCE, WINDOW and WRAP screen settings, a very useful way of showing how far off (or wrapped back onto) the screen your work is.

The default screen, as in Basic, allows re-definition of colour, margin, windows and backgrounds etc. These are essentially dependent on mode. All versions work in all modes though clearly graphics will not be attempted in 3,6, and 7. The handling of text and graphics etc. is compared in table 2.

The version supplied by LSL here began really to show its limitations - any resetting of the palette was actually undone after editing, presumably because the latter is done in mode 7 and thus resets your resetting! Unfortunately, all four versions clear your previous work from the screen on exit from the editor, which occupies the whole screen. It is a pity that you are unable to return straightaway to the graphics you had on the screen. Despite a persistent flicker during much of the graphics plotting, LSL's is the only version that allows variable graphics/text

## Table 1

| Features | Acorn | LSL | Logotron | BBC/OU |
|---|---|---|---|---|
| EP(ROM) price inc VAT | £69 | £67 34 | £67 78 | £69.95 |
| number of (EP)ROMs | 2 | 1 | 1 | 2 |
| Electron compatible | yes | version promised | no | no |
| Econet compatible | yes | yes — with patch | yes — with patch | yes |
| Floor turtles supported (1) | Jess/Val/Buggy | Jess/Val/Buggy | Jess/Val | BBC Buggy |
| Sprites | with Graphics ROM | 32 board soon | 30 board soon | no |
| Number of primitives | 150+ | 123 | 160 | 137 |
| Use of NODES | no | no | yes | yes |
| SAVE & LOAD a screen | yes | no | no | yes |
| Multiple Turtles | yes | no | no | yes |
| Detect a Key pressed | yes | no | yes | yes |
| SPOOL text on screen to Disc | no | yes | yes | no |
| Mathematical notation (2) | PRE & IN-fix | PRE-fix only | PRE & IN-fix | PRE & IN-fix |
| SAVE by Procedures on session | both | Procedures only | both | both |

*(1) Jessop, Valient, BBC Buggy (2) INFIX notation is "2+5"; PREFIX is "ADD 2 5"*

## Table 2

| Graphics | Acornsoft | LSL | Logotron | BBC/OU |
|---|---|---|---|---|
| Redefinable Turtle | yes | no | no | yes. easily saved |
| Fill command | by changing PLOT style | yes | by changing plot style | yes |
| Variable text windows | yes | yes: easy | only using VDU 24 & 28 | only using VDU 24 & 28 |
| Change palette | yes: easy | only using 2 commands | only by VDU 19 | yes |
| Different plotting styles | yes in a variety of ways | no | yes | yes |
| Detect a colour at a point | yes | no | yes | yes |
| Built in Printer Dump | yes - on extension disc | no | no | no |
| Write text in graphics area | yes | no - except by VDU 5 | no - except by VDU 5 | yes but see note (1) |
| Clear whole screen for text | yes | no | yes | yes |
| REM-type comments allowed | yes | yes | no | no |
| Upper & lower case accepted | both | upper only | upper only | both - see note (1) |

*(1) There is a bug that prevents one piece of text from overwriting another properly.*

windows including split-screens vertically without resorting to a VDU24/28 command. To change the palette with Logotron, it is necessary to use the VDU19 command or make a virtue out of necessity so:

```
TO SETPAL :A :B
MAKE "A [SE 19 :A :B 0 0 0]
VDU :A
END
```

There is much in a Logo environment to favour this building block approach. Memory permitting, a file of all such definitions can be created and then loaded at the start of each session.

Now compare this with the BBC/OU's COLOUR command with its two parameters. Most newcomers will prefer this at the beginning, experimenting with the intricacies of VDU statements only later. In this respect the FILL and PAINT commands of LSL and BBC/OU respectively must score over their rivals. The latter is exceptionally powerful (if slow), being akin to Basic's PLOT70. Acornsoft's Graphics ROM (available later this year) is also designed to interface with their Logo, increasing flexibility.

LIST PROCESSING

List processing combines string handling with features of Basic's arrays, but without the need either to distinguish between numbers and strings or dimension beforehand. It permits lists to be created effectively as variables and manipulated to exclude, for instance, all but their last item (be this a word, a number or a character). Thus Acornsoft's:

```
To reverse :text
if :text = " [output "]
output word (last :text)
            (reverse butlast :text)
end
```

would be called with:

```
print reverse "RECURSION
```

and would yield:

```
NOISRUCER
```

This recursive procedure appears more fearsome than it is! Don't be put off by the syntax (vital in exploiting Logo's precision), and its liking for the 'output' command. This invites the user to break a program down such that if a procedure returns a result, it is necessary actually to print it.

And yes, the immensely powerful technique of recursion - calling a procedure from within itself - is well catered for in all versions, but fastest in Logotron's.

GENERAL FEATURES

Table 3 assesses the editors, debugging facilities and program structure etc., but note the following:

1. It is a serious drawback that the LSL Editor requires lines to be typed in at the foot of the screen and entered (ZX81-like) with the Copy key. Any self-respecting version of Logo MUST have a proper screen editor. This does not!

2. Logotron has an excellent 'FIND & REPLACE' in the editor for text, but also uses function key f0 to delete a single character and f1 to delete to the end of line. If you have typed in a long line and press the wrong key...

3. The BBC/OU Editor is a very well intentioned but flawed screen editor

## Table 3

| Features | Acornsoft | LSL | Logotron | BBC/OU |
|---|---|---|---|---|
| Full screen Editor | ****(*) | N/A (1) | ***(*) (2) | *** (3) |
| Redefine Procedure name | 2 steps | very easy | inside editor | 2 steps |
| Tracing & Debugging | ***** | **** | * | *** |
| Trace variables' values | yes | yes | no | yes |
| Program structure: flexibility | ***** | ** | *** | ***** |
| Pause with Continue | yes | no | WAIT only | PAUSE only |
| speed (4) | *** | ** | ***** | * |
| LOGO error messages (5) | **(*) - lower | *(*) - lower | *** - upper | **** - lower |

too. It makes excellent use of colour but being in mode 7 (like LSL's) shows square brackets [ ] as arrows. There is also a disconcerting bug: if you take the TAB key beyond the text on your line, or move the up/down cursor keys beyond the edge of the screen, the text all disappears. It can easily be recovered but is a shock at first.

4. Much has been written about Logos' speeds of execution. Logotron scores too because of its superior rates in heavily recursive procedures.

5. It is preferable for children to read lower not upper case text. Imagine a book ALL IN CAPITAL LETTERS! The Open University version uses unconventional punctuation, which I found confusing.

DOCUMENTATION

The BBC/OU (I had only a provisional copy with no illustrations) was in keeping with their standard of thoroughness. In common with Acornsoft, it had an introductory 'tutorial' and a much more comprehensive reference section. BBC/OU also includes the best and longest (30 pages) 'hard-line technical' support section and an excellent introduction to data-types contributing persuasively to the contention that Logo IS a language for beginners! Acornsoft has a stand-alone reference card.

I found Logotron's manual hard to use because of its high, 'U'-shaped clip-lock binder, making turning its loose pages cumbersome. Yet this will make inclusion of later material as well as removal of pages to prop up at the keyboard easier. Swings and roundabouts again.

LSL, although the least adequate not even containing an index, was more obviously written with children in mind than the others.

Since children could not use the manuals unaided, it follows that, initially, someone who can, or who has experience of computer languages, will 'intervene'. It would thus be a false economy to claim that the less comprehensive two versions (LSL and Logotron) are necessarily any more accessible. It is impossible to produce the Logo that satisfied everyone, and made no concessions to speed for the sake of completeness. Flexibility thus counts for a lot more and weighed heavily in my overall conclusions.

If its speed you're after then at the top of your list you should put the single chip version by Logotron, who have also developed furthest their plans for future expansion.

If you go for the more 'complete' version (and I hope I have made it clear that there is no such thing as a 'full' Logo) then you should consider Acornsoft, which I confess I also preferred overall of the four.

Acornsoft,
Betjeman House,
104 Hills Road,
Cambridge CB2 1LQ.

BBC/OU, BBC Publications,
Software Department,
35 Marylebone High Street,
London W1M 4AA.
(schools use BBC Schools Order System)

Logotron Ltd,
5 Granby Street,
Loughborough LE11 3DU.
(schools order from E.J.Arnold)

Logo Software Ltd.,
316a Richmond Road,
Twickenham TW1 2PD.

Tested on O.S. 1.2
Basic I & II
6502 2nd proc.

# A SPREADSHEET PROGRAM (Part 2)

**We present part two of the BEEBUG Spreadsheet program. This provides the facility to save and load spreadsheets that you have previously created, and adds more sophistication and flexibility to the basic program published last month.**

ADDING PART TWO OF THE PROGRAM

The first task is to add the remaining program listed here to the basic program published last month. Type in and save part 2 in the usual way (you can leave out lines 10 to 60 as these are essentially the same as in part 1). You must also save a temporary spooled version of part 2. With part 2 already loaded into memory:

1. Type *SPOOL TEMP (or any other name of your choosing).
2. Type LIST to list and spool out the whole of part 2.
3. Type *SPOOL to complete this stage.

You can now combine parts 1 and 2:

4. Load the original part 1 program.
5. Delete lines 3380 to 3480 as these were only temporary.
6. Type *EXEC TEMP to add part 2.
7. Save the combined program under a suitable name (we shall refer to it as SPREADX), but keep parts 1 and 2 at least until the new program is fully tested.
8. You now have the complete SPREADX ready to use.

Should any errors be discovered, edit your original copy of part 2 and resave before repeating steps 1 to 8.

ADDITIONAL MENU OPTIONS

MENU OPTION 6 enables a spreadsheet previously saved to be loaded. You will find that the program now asks at the start of any run if an old or a new spreadsheet is to be used. A new spreadsheet can be loaded at any time.

MENU OPTION 7 allows you to save a spreadsheet, often the final action of a program run. By choosing different names you can easily save several different versions of the same spreadsheet, and using options 6 and 7 switch quickly between them.

MENU OPTION 8 provides hard copy output of both the specifications and the spreadsheet as selected by the user. The printout option can be readily tailored to your own printer. The program is set to print columns of 8 characters across an 80 character line (the values assigned to s% and w% in line 4950). Any printer control codes can be included in lines 5100, 5240, 5280 and 5340 to switch special effects on and off. If a spreadsheet will not fit the specified number of columns then it will be split up into two or more sections printed in sequence.

MENU OPTION 9 enables you to place the decimal point wherever you wish for any column of a spreadsheet. You follow the column letter prompt by the @% requirement as desired (see User Guide pages 70 and 327). For example, &20206 gives 2 decimal places in a six position field (the default is &20006).

SPECIFICATIONS (Menu Option 4)

Part 2 has added the full amendment and deletion routines to the basic spreadsheet program. Now, amending or deleting any specification will also ensure that any resulting calculated values in the table are also removed. The amend routine will display any selected specification for editing using Copy and the cursor keys.

When making any changes it is always advisable to show the existing specs first (option S). When any spec is deleted, subsequent specs are renumbered to close the gap. New specs may be entered at any position by specifying the appropriate index number, and the existing and subsequent specs will be moved up to make room. Remember that the order of your specifications is important in calculating the right results.

NEW SPECIFICATIONS

Last month's article explained:

1. Single box specs (e.g. AA=AB*AC) but be careful here for division by zero which cannot be easily trapped.

2. Totalling (e.g. AN=AB;AM).
3. Repeated totalling
(e.g.AN=AB;AM@D).

The full program additionally allows:

4. Operating on any row or column with a constant. Suppose row A column A contains your estimated weekly expenditure, row B (columns B to M) the number of weeks in each of the 12 months, and you want columns B to M of row A to contain the product of box AA times the number of weeks in each month. This can be achieved with:

AB.AM=BB*AA

The format AB.AM specifies the columns B to M inclusive of row A into which


are inserted the products of row B (starting from column B) and the constant in box AA. Any of the four operations (+, -, *, /) may be specified and the operation applied to either a row or a column. In the event of division by zero, the result is set to zero.

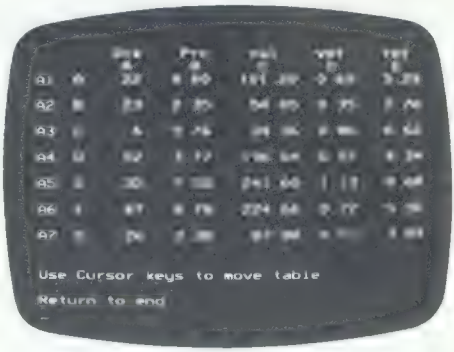
5. Calculating one row or column against another using the operations of +, -, *, or /. For example, column A (rows A to J) might contain stock levels for several items, column B similarly unit prices, and you wish column C to contain the corresponding stock values. The specification for this would be:

AC_JC=AA*AB

This means 'Into boxes AC to JC inclusive insert the products of AA (to JA) * AB (to JB)'. The range of columns is indicated by AC_JC (underline NOT minus). If you also added the spec:

KC=AC;JC (from last month)

you will produce the total value of the entire stock in box KC. We could also add two further specs:




AD.JD=AB*AK
AE_JE=AB+AD

If the current rate of VAT (Ø.15) is stored at AK (just for convenience) this would calculate the VAT on each item in column D and the total sale price for each item in column E. Any division by zero again gives a zero.

ADDITIONAL COMMENTS

Always ensure that any model you create is of adequate size by using sufficient rows and columns initially as these cannot be changed later.

It is not essential when using ranges of rows or columns in specs that they should use the same letters, such as when the values in one column (say A to J) are multiplied by the values in another column, and the results inserted in yet a further column. You can equally well use any similar number of consecutive cells. for example AA_JA=BB*CC would insert into column A (rows A to J) the product of column B (rows B to K) and column C (rows C to L). This permits great flexibility.

For example, suppose a spreadsheet is to show a twelve monthly annual bank balance. If, say, row B contains payments, and row D contains receipts, then the specification:

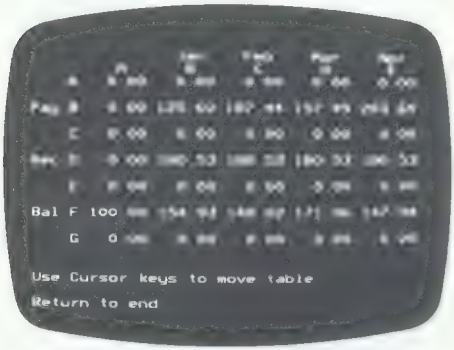
FB_FM=DB-BB

would put in row F (columns B to M) the net receipts. Then, provided that FA contained the opening balance, the spec

FB_FM=FB+FA

would calculate correctly, month by month, the balance.

Although using the Spreadsheet Program may seem complicated at first,




you will find with practice that quite complex spreadsheets can be readily developed. Work through the examples given, try experimenting further and you will soon begin to see why spreadsheets are considered one of the most useful applications ever to find its way onto a computer.

We have produced a comprehensive information leaflet on the Spreadsheet Program that is available free of charge if you send an A5 (or larger) SAE. This contains detailed instructions on the use of the Spreadsheet Program, and comprehensive Program Notes for those who might wish to extend or modify the program for themselves. We have also included another useful example using the Spreadsheet Program. Send for your leaflet to Spreadsheet Facts, BEEBUG, P.O.Box 50, St Albans, Herts.

The full Spreadsheet Program (combining parts 1 and 2) is also included on this month's magazine cassette/disc.

SOME PROGRAM NOTES

These notes provide a brief but useful introduction to some of the more important attributes of the Spreadsheet Program as a whole.

MODEL SIZE

The procedure PROCinit (from line 3240 onwards) sets up the main arrays used by the program. The three main arrays (mat, col$ and row$) are dimensioned to the maximum possible size, but any model is limited to the size in rows and columns (values of y% and x% respectively) specified when that model was first set up. This keeps calculation to a minimum and also reduces storage requirements (and time) when saving and loading spreadsheets. The number of specifications is limited to 100 by the dimensioned size of array Spec$, but this could be increased.

VALIDATION

With part two in operation, every specification is checked by the procedure PROCvalidate (line 5550 onwards). In principle all checking takes place here - calculations assume correct specifications. The procedure divides specifications into three types

Simple expressions (5590 - 5610)
Summation and repeated summation (5620 - 5700)
Operations on rows and columns (5710 - 5750)

The specification to be checked is stored in a user defined area of memory (see line 3275) and accessed using string indirection operators. The level of checking provided is limited but could be readily extended.

ERROR TRAPPING

Error trapping (see BEEBUG Workshop Vol.3 No.9) can cause problems in any program which makes extensive use of functions and procedures. Should the program be terminated prematurely it will normally be possible to continue by using the GOTO instruction in immediate mode as specified on the screen at the time.

The most likely causes of error are when saving and loading spreadsheets (file missing, no room, etc) and when making calculations (menu option 5) due to errors or inconsistencies not detected by the version of PROCvalidate provided. When developing any spreadsheet it is best to save copies at frequent intervals as you proceed.

```
  10 REM Program SPREADX
  20 REM Author A.BEEBER
  30 REM Version 1.9L/2
  40 REM BEEBUG APRIL 1985
  50 REM Program subject to Copyright
  60 :
 190 IFoption%=6 THEN PROCload
 200 IFoption%=7 THEN PROCsave
 210 IFoption%=8 THEN PROChardcopy
 220 IFoption%=9 THEN PROCedit@
1712 PROCvalidate(sp$)
```

```
 1714 IFerror% PRINT"Bad spec.":GOTO178
Ø
 2420 IFMID$(Spec$(K%),3,1)="." THENPRO
Cmultiply(Spec$(K%),1):GOTO2490
 2430 IFMID$(Spec$(K%),3,1)="_" THENPRO
Cmultiply(Spec$(K%),2):GOTO2490
 3275 DIM spc% 20
 3280 IF FNcont(Ø,1,"Is this a new spre
adsheet (Y/N)?","YN")="N" THEN PROCload
:GOTO 3330
 4000 DEFPROCsave
 4010 CLS:PRINT"Saving data file"
 4020 INPUT"Enter a filename"'"(Max.6 c
haracters):"Dfile$
 4030 A$=FNcont(POS,VPOS+1,"Insert data
 disc and press Return",CHR$13)
 4040 PRINT"Please wait"
 4050 F%=OPENOUT Dfile$
 4060 PRINT#F%,x%,y%
 4070 FORI%=ØTO 200
 4080 PRINT#F%,Spec$(I%)
 4090 NEXT
 4100 FORI%=ØTO(y%-1)
 4110 FORJ%=ØTO(x%-1)
 4120 PRINT#F%,mat(I%,J%)
 4130 NEXT,
 4140 FORI%=ØTOy%-1:PRINT#F%,row$(I%):N
EXT
 4150 FORI%=ØTOx%-1:PRINT#F%,col$(I%),e
dit%(I%):NEXT
 4160 CLOSE#F%
 4170 ENDPROC
 4180 :
 4190 DEFPROCload
 4200 CLS:PRINT"Loading data from file"
 4210 INPUT"Enter a filename"'"(Max.6 c
haracters):"Dfile$
 4220 A$=FNcont(POS,VPOS+1,"Please inse
rt data disc and press Return",CHR$13)
 4230 F%=OPENUP Dfile$
 4240 PRINT"Please wait":M%=-1
 4250 INPUT#F%,x%,y%
 4260 z%=x%*y%:col%=Ø:row%=Ø
 4270 FORI%=ØTO 200
 4280 INPUT#F%,Spec$(I%)
 4290 IFSpec$(I%)="" AND M%<Ø M%=I%
 4300 NEXT
 4310 FORI%=ØTO(y%-1)
 4320 FORJ%=ØTO(x%-1)
 4330 INPUT#F%,mat(I%,J%)
 4340 NEXT,
 4350 FORI%=ØTOy%-1:INPUT#F%,row$(I%):N
EXT
 4360 FORI%=ØTOx%-1:INPUT#F%,col$(I%),e
dit%(I%):NEXT
 4370 CLOSE#F%
 4380 ENDPROC
 4390 :
 4400 DEFPROCam
 4410 LOCAL I%
 4420 CLS:PRINT"Index";SPC6;"Specifications"
 4430 REPEAT
 4440 REPEAT:I%=FNinput(POS,VPOS,3,"I")
:UNTIL I%<M% AND NOT NUL%
 4450 PRINTTAB(10,VPOS);Spec$(I%)
 4460 sp$=FNinput(10,VPOS,19,"S")
 4470 IF NUL% THEN 4520
 4480 PROCvalidate(sp$)
 4490 IF error% THEN PRINT"Bad spec.":G
OTO4520
 4500 PROCclean(Spec$(I%))
 4510 Spec$(I%)=sp$:PRINT
 4520 G$=FNcont(Ø,19,"Space to continue
 - Return to exit",CHR$32+CHR$13)
 4530 UNTIL G$=CHR$13
 4540 ENDPROC
 4550 :
 4560 DEFPROCde
 4570 CLS:PRINT"Index"
 4580 REPEAT
 4590 I%=FNinput(POS,VPOS,3,"I")
 4600 IF NUL% OR I%>M%-1 THEN 4650
 4610 PRINTTAB(10,VPOS);Spec$(I%)
 4620 PROCclean(Spec$(I%))
 4630 Spec$(I%)="":M%=M%-1
 4640 FORJ%=I%TOM%:Spec$(J%)=Spec$(J%+1
):NEXT
 4650 G$=FNcont(Ø,20,"Space to continue
 - Return to exit",CHR$32+CHR$13)
 4660 UNTIL G$=CHR$13
 4670 ENDPROC
 4680 :
 4690 DEFPROCclean(A$)
 4700 $(spc%+1)=A$
 4710 V%=spc%?1-B%:Z%=spc%?2-B%:w1%=spc
%?3
 4720 V1%=spc%?4-B%:Z1%=spc%?5-B%:w2%=I
NSTR(A$,"@")
 4730 IFw2%>ØTHEN PROCrepeat ELSE IFw1%
=46ORw1%=95THENPROCmulti ELSEmat(V%,Z%)
=Ø
 4740 ENDPROC
 4750 :
 4760 DEFPROCrepeat
 4770 w1%=spc%?10-B%
 4780 IFZ%=Z1% THEN V1%=w1%:PROCzerocol
 ELSE Z1%=w1%:PROCzerorow
 4790 ENDPROC
 4800 :
 4810 DEFPROCmulti
 4820 IF V%=V1% THEN PROCzerorow ELSE P
ROCzerocol
 4830 ENDPROC
 4840 :
 4850 DEFPROCzerocol
 4860 FORJ%=V%TOV1%:mat(J%,Z%)=Ø:NEXT
 4870 ENDPROC
 4880 :
 4890 DEFPROCzerorow
 4900 FORK%=Z%TOZ1%:mat(V%,K%)=Ø:NEXT
 4910 ENDPROC
 4920 :                                  ▶
```

```
 4930 DEFPROChardcopy
 4940 CLS:PRINT"Hard copy"'
 4950 C%=0:s%=8:w%=80
 4960 PRINT"Enter 1. for details"'SPC6;
"2. for specs"'SPC6;"3. for BOTH"
 4970 REPEAT G%=GET:UNTILG%>48AND G%<52
 4980 D1%=((w%-6)DIV(s%))-1:D2%=w%DIV24
 4990 IFG%=50THENPROCprintspecs:ENDPROC
 5000 REPEAT
 5010 IFD1%<x%-C%THENcc%=D1% ELSEcc%=x%
-C%-1
 5020 PROCprinter(cc%,y%-1)
 5030 C%=C%+D1%+1:cc%=x%-C%-1
 5040 UNTIL C%>=x%
 5050 IFG%=51THENPROCprintspecs
 5060 ENDPROC
 5070 :
 5080 DEF PROCprinter(co%,ro%)
 5090 LOCALI%,J%,K%:J%=C%
 5100 VDU2:REM Any printer codes here
 5110 PRINT'"Spreadsheet ";Dfile$'
 5120 FORI%=0TOco%
 5130 PRINTTAB(8*I%+9);col$(I%+J%);
 5140 NEXT:PRINT
 5150 FORI%=0TOco%
 5160 PRINTTAB(8*I%+10);CHR$(B%+I%+J%);
 5170 NEXT:PRINT'SPC1
 5180 FORI%=0TOro%
 5190 PRINTrow$(I%);TAB(4);CHR$(B%+I%);
 5200 FORK%=0TOco%
 5210 @%=(edit%(C%+K%)AND&FFFF0)+s%:PRI
NTmat(I%,C%+K%);
 5220 NEXT:PRINT
 5230 NEXT
 5240 VDU1,12,3:REM any printer codes
 5250 ENDPROC
 5260 :
 5270 DEF PROCprintspecs
 5280 VDU2:REM printer codes here
 5290 PRINT'"Specifications ";Dfile$'
 5300 FORI%=0TOM%-1
 5310 J%=24*(I%MOD D2%)
 5320 PRINTTAB(J%);I%;TAB(J%+4);Spec$(I
%);
 5330 NEXT:PRINT
 5340 VDU1,12,3:REM any printer codes
 5350 ENDPROC
 5360 :
 5370 DEFPROCmultiply(A$,T%)
 5380 LOCAL A1%,A2%,B1%,B2%,C1%,C2%,D1%
,D2%,Z$,Z1$:Z$=MID$(A$,9,1):Z1$=""
 5390 $(spc%+1)=A$
 5400 A1%=spc%?1-65:A2%=spc%?2-65
 5410 B1%=spc%?4-65:B2%=spc%?5-65
 5420 C1%=spc%?7-65:C2%=spc%?8-65
 5430 D1%=spc%?10-65:D2%=spc%?11-65
 5440 IF T%=2 THEN Z1$="+I%"
 5450 IF A1%=B1% THEN 5500
 5460 FORI%=0TO(B1%-A1%)
 5470 IFmat(D1%+I%,D2%)=0 AND Z$="/" TH
EN mat(A1%+I%,A2%)=0:GOTO5490
 5480 mat(A1%+I%,A2%)=EVAL("mat(C1%+I%,
C2%)"+Z$+"mat(D1%"+Z1$+",D2%)")
 5490 NEXT:ENDPROC
 5500 FORI%=0TO(B2%-A2%)
 5510 IFmat(D1%,D2%+I%)=0 AND Z$="/" TH
ENmat(A1%,A2%+I%)=0:GOTO5530
 5520 mat(A1%,A2%+I%)=EVAL("mat(C1%,C2%
+I%)"+Z$+"mat(D1%,D2%"+Z1$+")")
 5530 NEXT:ENDPROC
 5540 :
 5550 DEFPROCvalidate(A$)
 5560 LOCALix%:$(spc%+1)=A$:error%=0
 5570 IF(spc%?3=46ORspc%?3=95)ANDspc%?6
=61THEN5710
 5580 IFspc%?3=61ANDspc%?6=59THEN5620
 5590 IFspc%?3<>61THEN5760
 5600 IFspc%?1>ASC(maxrow$)THEN5760
 5610 IFspc%?2>ASC(maxcol$)THEN5760ELSE
5750
 5620 IFFNval1($spc%)THEN5760
 5630 IFLEN(A$)<9THEN5750
 5640 IFspc%?9<>64THEN5760
 5650 IFspc%?1=spc%?4THEN5690
 5660 IFspc%?2<>spc%?5THEN5760
 5670 IFspc%?10>ASC(maxcol$)THEN5760
 5680 ENDPROC
 5690 IFspc%?10>ASC(maxrow$)THEN5760
 5700 ENDPROC
 5710 IF NOT(spc%?9=47ORspc%?9=45ORspc%
?9=43ORspc%?9=42)THEN5760
 5720 IFspc%?10>ASC(maxrow$)THEN5760
 5730 IFspc%?11>ASC(maxcol$)THEN5760
 5740 IFspc%?3=46AND(spc%?5-spc%?2+spc%
?11)>ASC(maxrow$)THEN5760
 5750 ENDPROC
 5760 error%=-1:ENDPROC
 5770 :
 5780 DEF FNval1($spc%)
 5790 FORix%=1TO7STEP3
 5800 IFspc%?ix%>ASC(maxrow$) OR spc%?i
x%<65THENerror%=-1
 5810 NEXT
 5820 FORix%=2TO8STEP3
 5830 IFspc%?ix%>ASC(maxcol$) OR spc%?i
x%<65THENerror%=-1
 5840 NEXT
 5850 =error%
 5860 :
 5870 DEFPROCedit@
 5880 CLS:PRINT"Editing @%"
 5890 PRINTTAB(0,4);"Enter @% column by
 column, (@ to end)."
 5900 FORI%=0TOx%-1:PRINTCHR$(I%+65)+"
&";:A$=FNinput(POS,VPOS,6,"S")
 5910 IFA$="@"THENI%=x%:GOTO5940
 5920 IFNUL% THEN5940
 5930 edit%(I%)=EVAL("&"+A$)
 5940 NEXT
 5950 ENDPROC
```

# BEEBUG Workshop

Tested on O.S. 1.2 Basic I & II

## SEARCHING AND SORTING (Part 1)

By Surac

**Searching and sorting are processes fundamental to many applications. Surac looks at some of the more useful techniques that can make searching and sorting of data faster and more efficient.**

From time to time most programmers need to sort a list of items into order. Maybe it's a set of scores, or perhaps a list of names to be put into alphabetical sequence. This month I'll give details of a couple of straightforward methods and suggest code which you could use in your own programs.

First, we'll look at the well-known and aptly-named "Bubble Sort". Suppose we must put a list into ascending order. The bubble sort starts with the first 2 elements, compares them and, if needed, swaps them so that the larger is in position 2. It then compares elements 2 and 3 and again puts the larger value into the higher position.

```
                BUBBLE SORT
10000 DEF PROCbubble(ST%,FIN%)
10010 IF ST%>=FIN% THEN ENDPROC
10020 LOCAL F%,I%
10030 REPEAT
10040   F%=FALSE
10050   FOR I%=ST% TO FIN%-1
10060     IF array(I%)>array(I%+1) THEN
          PROCswap
10070     NEXT
10080   FIN%=FIN%-1
10090   UNTIL NOT F%
10100 ENDPROC
10490:
10500 DEF PROCswap
10510 LOCAL temp
10520 temp=array(I%)
10530 array(I%)=array(I%+1)
10540 array(I%+1)=temp
10550 F%=TRUE
10560 ENDPROC
```

The sort continues until it reaches the end of the list when, all being well, the largest element will have reached the top. It has "bubbled" up through the list. The sort then goes back to the start and bubbles the next-largest element up to the second from top position. So it goes on until the whole list is sorted.

If there is much swapping to do, the bubble sort can be painfully slow. However, as soon as a pass through the list is made without swapping anything, the whole lot is then sorted. This means that, with only a few items out of place, the sort can be very fast indeed.

The procedure PROCbubble assumes that the data to be sorted is in "array()". Obviously, you should use your own variable name here. The routine expects two input parameters: ST%, which defines the first element of the array to be sorted, and FIN% which defines the last. This means that you don't have to sort an entire array every time. For instance, if array() had 300 elements, PROCbubble(100,200) would sort the middle third only. The procedure makes sure the limits are sensible. The subsidiary procedure PROCswap swaps two elements when needed.

After each pass through the array, we know that the next highest value has reached its final position; FIN% is thus reduced by 1 so that we don't waste time checking the sorted items at the top of the array. F% shows if there are any swaps in a pass through the list, and allows an early exit.

The bubble sort is simple but can be slow. However, there is a much faster version known as the 'Shell Sort' after its originator. This time, instead of always comparing adjacent elements, the sort starts by comparing, and swapping, items which are separated by some distance. Whenever no swaps occur in a pass, this distance is halved and the sorting starts again.

The process continues until the gap is 1, when it is just like a bubble sort. However, by the time it gets there, the list has already been sorted into rough order and the whole thing finishes very quickly. There is another procedure to do this job.

```
                SHELL SORT
11000 DEF PROCshell(ST%,FIN%)
11010 IF ST%>=FIN% THEN ENDPROC
11020 LOCAL F%,I%,S%,T%
11030 S%=2^INT(LOG(FIN%-ST%)/LOG(2))
11040 REPEAT
11050   T%=FIN%-S%
11060   REPEAT
11070     F%=FALSE
11080     FOR I%=ST% TO T%
11090       IF array(I%)>array(I%+S%)
            THEN PROCswaps
11100       NEXT
11110     T%=T%-1
11120     UNTIL NOT F%
11130   S%=S% DIV 2
11140   UNTIL S%=0
11150 ENDPROC
11490:
11500 DEF PROCswaps
11510 LOCAL temp
11520 temp=array(I%)
11530 array(I%)=array(I%+S%)
11540 array(I%+S%)=temp
11550 F%=TRUE
11560 ENDPROC
```

You can see its links with the bubble sort. At line 11030, S% is set to the initial gap value. This must be a power of 2 (so it can be continually halved as the sort progresses) and the line calculates the largest number that will fit between ST% and FIN%. T% holds the upper limit of the FOR-NEXT loop; its start value is set so that the program does not go outside the array and, as before, it is decremented on every pass.

It's hard to say how much better the Shell sort is than the bubble, since so much depends on the starting data. In general terms, though, the bigger the array, the relatively faster it is: 200 random elements are sorted about 4 times quicker, while 500 gives an advantage of around 7. It is NOT always quicker though. If you are certain that only a few - say no more than 2% - of the items in a list are misplaced, then use the bubble sort. It will probably correct them all in a single pass, whereas the Shell sort must always have at least one pass at each gap setting.

Finally, let's have a look at these two sorts in action with this code:

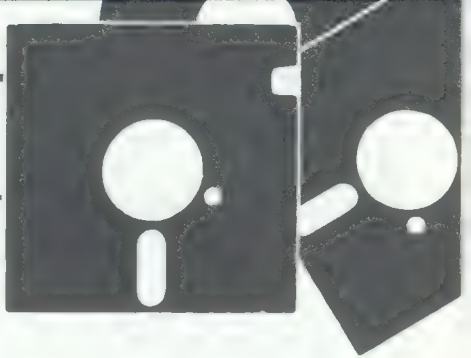
```
 10 MODE7
 20 P%=HIMEM+159
 30 PROCfill
 40 PRINT TAB(0,1)"Bubble Sort:"
 50 TIME=0
 60 PROCbubble(1,200)
 70 TBUB=TIME
 80 CLS
 90 PROCfill
100 PRINT TAB(0,1)"Shell Sort: "
110 TIME=0
120 PROCshell(1,200)
130 TSHL=TIME
140 PRINT TAB(5,18) "Bubble sort: ";
    TBUB/100;" secs"
150 PRINT TAB(5,20) "Shell sort: ";
    TSHL/100;" secs"
160 END
990:
1000 DEF PROCfill
1010 FOR I%=1 TO 200
1020   P%?I%=64+RND(26)
1030   NEXT
1040 ENDPROC
```

Add the two sort routines, changing every occurence of array (I%) or array (J%+S%) to P%?I% or P%?(I%+S%) respectively. Other references to arrays should similarly be changed.

Run the program and two random 200-element byte arrays are created and sorted. However, since P% points to the mode 7 screen memory, the data is displayed on the screen as characters which you can see being put into order. If you increase to, say, 500 bytes rather than 200, you will see just how much the sorts slow down.

You can, of course, play all sorts of variations on this theme. Try watching the effect of bubble and Shell sorts of arrays with only one element misplaced. Try it with everything starting in reverse order. How would you change the sorts to give the result in descending order?

# UNDERSTANDING DISC FORMATTING



**If you have ever suffered by having to convert between 40 and 80 track drives. then the idea of a switchable 40/80 track unit might seem the answer. James Fletcher explains clearly exactly what's going on when you format a disc and shows why all is not always what it seems.**

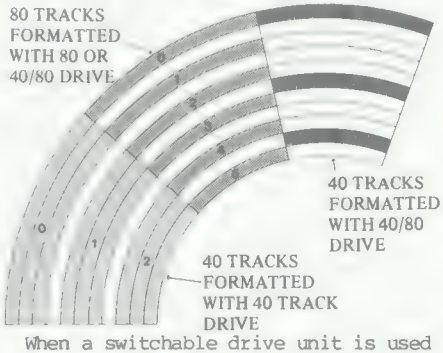
If you have ever encountered problems in using the same discs on standard 40 track disc drives and on switchable 40/80 track drives then my experiences should throw some light and understanding on this vexing situation.

I recently decided that I would replace my old 40 track Acorn disc drive unit with a double sided 80 track twin unit, so as to allow myself much more storage capacity and much easier copying from drive to drive. Knowing that some of my friends still had 40 track drives I made sure that the new drives were switchable between 40 and 80 tracks, so that I could cope with all eventualities. I managed to find a unit with the track-change switches on the front; many drives have the switches at the back which makes life difficult if, like mine, the computer is built into a console.

Using the new unit was a joy, and it didn't take long for the contents of about 40 discs to be copied onto 10 double sided 80 track discs. However, it was when I next took some of my original software masterpieces to work to show off to my friends that the problems started. Using the new switchable drives I copied some programs onto 40 track discs suitable for the drives at work. When these were put into the 40 track drives the only response was 'Drive fault 18 at 00/00', which was annoying, to say the least. Other drive units were tried with similar results, and a good deal of head scratching (mine, not the drive's!) followed.

The Disc System User Guide proved as useless as ever, and a detailed perusal of the official data sheet for the 8271 disc controller chip provided only the fascinating information that my problems were due to 'bad track' errors. I then started to delve into the workings of disc drives and came up with the simple explanation for my problems.

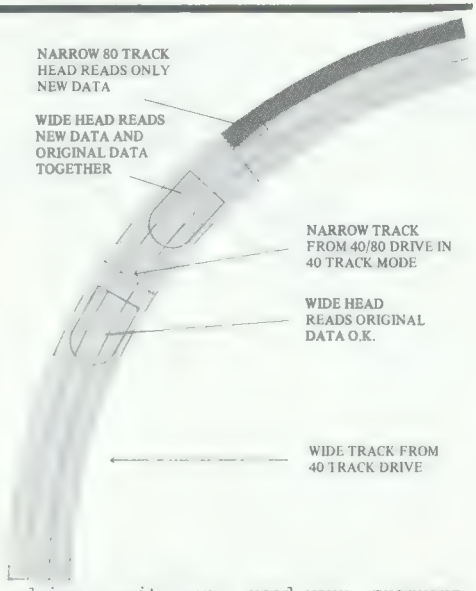
It turns out that 'standard' 40 track disc drives such as my Acorn original have read/write heads that are physically twice as wide as the heads on 80 track machines, so that they lay down formatting tracks on the discs that are twice as wide as those laid down by the narrower 80 track heads. The diagram shows what happens, and makes the reasons for my difficulties plain.




When a switchable drive unit is used in the 40 track mode it is made to jump two tracks at a time and so does, of course, write forty tracks, but these are narrow tracks. If another narrow-headed (switchable 80 track) drive is used it will be able to read the 40 tracks without difficulty. If the disc that is being used was formatted on a '40 track narrow' machine, again there will be no problems, since even if this disc is read on a '40 track wide' unit the wide head will read only the information that has been laid down by the narrow head that went before.

Purists say that even this situation is undesirable because the wide head will be bound to pick up less signal information from a narrow track than from a wide one, and will be therefore more likely to give errors. In practice I have never found this to be a source of problems, although if you are a wide 40 track drive user using discs that have been created on a narrow track unit it would probably be sensible to make wide track backups on your own machine.

The real problems, the ones that I encountered , come about when you use a wide head drive to try to read a disc that has been written to on a wide head machine but subsequently written to by a narrow head. Figure 2 shows the situation.




The wide head reads both the wanted narrow track that you have just recorded and the half of the track's width that remains from previous write operations. This means that the head reads a mixture of both old and new signals, which provides your computer with a garbled signal that it cannot possibly decipher, so it is not surprising that error messages result.

If you want to be quite sure that a friend with a 40 track, wide head drive unit can read your programs that have been saved using a 40/80 track narrow head drive you should copy them onto a brand new disc that has never been written to with a wide head. Once a disc has had wide tracks laid down on it no amount of reformatting with a narrow head unit will clear the tracks completely.

# ★Special Offer★Competition★

We are pleased to be able to offer the AMX Mouse package (reviewed earlier in this issue) to BEEBUG members at a special reduced price. The complete package, consisting of the mouse, control ROM, AMX Art program, Design program, and manuals, is available to BEEBUG members only for £79.95 inclusive of postage and packing, and VAT - a saving of £10 over the usual price. Overseas members should submit the same amount,as the VAT portion of this price covers the extra postage costs.

All orders for this offer should be sent to the subscription address at High Wycombe. See the supplement for full ordering details.

COMPETITION

If you fancy an AMX Mouse for absolutely nothing then we have a competition for you. It is obvious that the AMX Mouse has great potential. What we want you to do is to come up with an idea that will realise some of that potential. Send us a description of a program, or even a suite of programs, that you think would bring out the best in the AMX Mouse. The most imaginative and useful idea will win a complete AMX Mouse package donated by AMS. Make your description concise but detailed. Read the review in this issue carefully and try to think of an idea that really makes use of the mouse's unique features. Your application should do the job better than is possible without the mouse.

The closing date for entries is the 30th April. Send your ideas to the Editorial address clearly marking the envelope 'Mouse Competition'.

Tested on O.S. 1.2
Basic I & II
6502 2nd proc.

# MAKING MUSIC ON THE BEEB (Part 3)

**Now that all the basic ideas have been covered, Ian Waugh starts putting theory into practice with some of the more interesting musical applications.**

This month's music article from the author of "Making Music on the BBC Computer" looks at a method of programming multi-part tunes.

Single-part tunes using only one sound channel are fairly easy to program. We can use a simple loop such as this:

```
10 FOR note=1 TO numberofnotes
20 READ env,pitch,dur
30 SOUND 1,env,pitch,dur
40 NEXT note
50 DATA E1,P1,D1,E1,P2,D2,E3,P3,D3
60 DATA.....etc
```

Most readers will probably have experimented along these lines and the first program to accompany these articles in BEEBUG Vol.3 No.8 played a single channel version of Mozart's Rondo Alla Turca. When we come to consider playing two, three or all four channels together we run into the problem of synchronization or how to keep the channels together. We saw last month how the sync command (S) in the SOUND statement (SOUND &HSFC,A,P,D) can be used to ensure two or more channels sound at exactly the same time. To make the most of this command we must also ensure that the note data is presented to the SOUND statements in a convenient order. We'll see why now.

QUEUES AND BUFFERS

The BBC micro uses a system of queues in its handling of sound information. This is how a program can seem to 'run ahead' of the SOUND statements it contains. It can be demonstrated as follows:

```
10 FOR N=53 TO 73 STEP 4
20 SOUND 1,-15,N,10
30 NEXT
40 PRINT "Finished!"
```

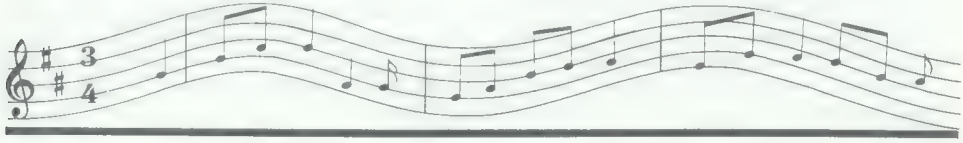
When run, "Finished!" appears on the screen immediately and the sounds follow on. As they are playing you can list the program and even make alterations to it. Alter line 10 to:

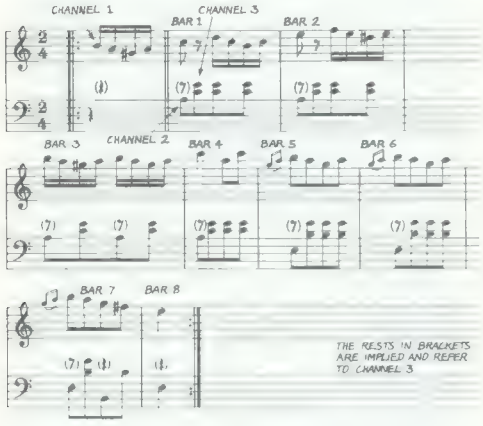
```
10 FOR N=53 TO 77 STEP 4
```

and you'll see that this time "Finished!" doesn't appear until after the first sound. This is what happens: when the computer comes across a SOUND command it puts the note information into a storage area or buffer. If the sound generator is empty, i.e. not playing a note, then a sound is sent to it from the buffer. When that sound has completed, another one is taken from the buffer and all the remaining notes move up a place. This is known as a first-in-first-out arrangement and can be likened to people queuing in a shop. When the person at the head of the queue is served everyone else moves up one position. Newcomers go to the back of the queue. Basic's job is over once it has put the information into the buffer and the program can carry on while the sound generator processes the sound information. When the buffer fills, however, the program is held up waiting for a free space - and this is what happens in the last example. There are actually four buffers, one for each channel and each can hold five sounds.

THREE-PART TROUBLE

Let's consider a practical application. We'll use Mozart's Rondo Alla Turca again, this time we'll produce a three-channel version. Look at the notation. If your music theory is a little shaky you may get some help from the first article in this series, mentioned above, although you don't need to be able to read music to understand the problem invloved.






The most obvious arrangement is to allocate a channel to each of the three parts. You can see from the figure, however, that by the time we reach the end of bar three, 24 notes will have passed through channel 1 and only eight through channel 3. If we try to send the information note for note, channel 1 would play quite merrily while channel 3's buffer filled with its longer notes. The program would seize up when its buffer was full. If the note lengths of all the parts are roughly equal, you may be able to get by with data arranged like this:

```
DATA Chan1,E1,P1,D1,Chan2,E2,P2,D2
DATA Chan1,E3,P3,D3,Chan2,E4,P4,D4
```

but not in many instances and you would certainly find it very restrictive.

USING ARRAYS AND THE ADVAL FUNCTION

The answer is to fill arrays, one for each channel, with the required note information. This can be read from the array when the buffer can take it without holding up the program. We can determine how full a buffer is by using the ADVAL function with a negative argument. This information is hidden away on page 204 of the User Guide. It returns the number of free spaces in the sound buffers. ADVAL(-5) checks channel 0, ADVAL(-6) checks channel 1, ADVAL(-7) checks channel 2 and ADVAL(-8) checks channel 3. Its use can be demonstrated by inserting this line in the above examples:

```
15 PRINT ADVAL(-6)
```

This prints the number of free spaces in the buffer and you will see how, as the program loops, the spaces fill up. The results returned by the function can be misleading. Just one sound in the buffer will return 12 and an empty channel will return 15. The space in the buffer available for notes is a third of these values so you may wish to divide the number by 3. For our purposes we only need to know if we can squeeze another note into the buffer and for this purpose something along these lines:

```
REPEAT
IF ADVAL(-6)>0 THEN SOUND 1,E,P,D
IF ADVAL(-7)>0 THEN SOUND 2,E,P,D
IF ADVAL(-8)>0 THEN SOUND 3,E,P,D
UNTIL finished
```

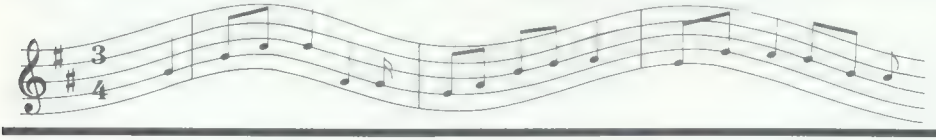
will suffice.

ADDING MORE PARAMETERS

You will have noticed in these examples that I have been using the variable, E, to represent an envelope with the implied assumption that each note could be allocated a different one. We must specify pitch and duration and preferably a sync parameter. Adding envelopes means specifying four parameters per note, not that this would be at all difficult, only time consuming. In practice you will rarely want to use a different envelope for each note although using different envelopes for different sections of the music is very effective. I usually program the envelopes separately (see the program) so cutting down on the amount of individual data required.

SYNCHRONIZING THE CHANNELS

We also need to specify whether or not there are any 'special' conditions attached to the sound such as sync,



hold or flush. It aids readability and debugging if data is entered in lines of one bar. We don't need to sync every note but we can conveniently sync notes at the start of each bar. Sometimes that may not be possible, for example if a note is tied or held over from a previous bar. In such cases the rule is sync where you can. Actually, tunes will probably stay fairly well in time without much sync but, as we saw last month, if the notes are synced together we can send Basic off to do something else while the music is playing. More about this next month. For the time being, we can print out the state of channel 1 by adding this line:

```
715 PRINT Chan1(1,Ch1)+1 TAB(4)Chan1
(2,Ch1) TAB(9)Chan1(3,Ch1) TAB(14)Chan1
(4,Ch1)*Tempo
```

If the channels were not synchronized, the delay between channel 1 and channel 2 caused by Basic taking time out to process this command would throw the program out of sync. The program allows us to give any individual note an attribute simply by preceding the note data with the attribute in hex form, i.e. preceded by "&".

It's time to put the theory into practice. After entering and running the program you can replay the tune by entering GOTO 680 rather than wait for it to analyse the data again. You can also alter Tempo in command mode at this time (see line 260). The book contains data and information for playing a further 24 bars of Rondo.

ENTERING YOUR OWN TUNES

If you're wondering why most printed examples of computer programmed music are based on the classics and not Boy George, Duran Duran or Tom Dolby the main reason is probably one of copyright. However, with this program you should be able to play many other 3-part tunes by inserting new data and altering the variables C1, C2 and C3. Alter the assignment of Env, too.

An easy mistake to make is to insert wrong C1, C2 or C3 values which will cause a channel to fill with another channel's notes. You could add more error checks (than those provided in PROCAnalyseNote) by inserting a termination character at the end of the data: so for example, if Note$ read a "*" and N did not equal C1 you would know the data was incorrect or C1 had the wrong value.

From this program it should be fairly easy to add a rhythm track using channel 0.

The figure and program are from Making Music on the BBC Computer by Ian Waugh, published by Sunshine Books at £5.95 and used with kind permission of the publishers.

PROGRAM NOTES

The number of notes in each channel is assigned to the variables C1, C2 and C3. These numbers need to be accessed several times during the course of the program so if you insert new data you need only alter these values once. The arrays at lines 200 to 220 are DIMensioned to hold information about each note.

The next section analyses the data and puts the resulting figures into the arrays as can be seen in lines 350, 370, 390 and 400. The process is repeated once for each channel. Although some clever programming could probably reduce the length of the code I have kept it this way to aid understanding. It is also easy to substitute new tune information and generally customise the program to your own needs. As the process is exactly the same for each channel we will only look at channel 1 in detail.

The FOR/NEXT loop between lines 330 and 410 runs through the data, once for each note. The first call is to PROCChan which 'cautiously' examines the first data item. If this begins with an ampersand (&) it knows it's a

channel instruction and evaluates the string with EVAL to produce the channel attribute, Chan. It then proceeds to read Note$ and Duration. Otherwise, it assumes Note$ is a note and reads Duration. In this case Chan is set to 0. This method of assigning attributes saves us having to enter an attribute for every note. We only enter one if we need one, making sure to precede it with an ampersand. Line 360 is my way of programming envelope changes: it's quick and simple.

PROCAnalyseNote has one difference to the version in the first part of this series in BEEBUG Vol.3 No.8 and that is in the way it handles a rest. If Note$ is an "R" Env is set to 0 in line 360 and the "R" is just used to exit from the analysis procedure. Pitch is arbitrarily set to 255. It shouldn't be heard but if it is you know something's gone wrong.

At the completion of these sections the arrays will be filled with data the SOUND command can work on directly although we can still modify them if we wish. The next routine plays the tune.

Ch1, Ch2 and Ch3 count the number of notes sent to each channel and the routine ends when all the notes have been sent - see line 740. The REPEAT loop between lines 700 and 740 does the work. Again, the principle for each channel is the same so we will only look at channel 1. First, the buffer is checked to see if it has space for another note and Ch1 is compared to C1 to see if channel 1 has had its allotted notes. If there is space and there are more notes to come Ch1 is incremented and used to access the data in the Chan1 array. It is at this point that the channel number is added (although it could also have been added in line 350). Here also, the duration is multiplied by the variable, Tempo.

Note that the program listing contains many comment lines (REM statements) to assist understanding, though these can be omitted when typing the program in.

```
  10 REM PROGRAM MUSIC9.2
  20 REM VERSION B0.1
  30 REM AUTHOR  I.Waugh
  40 REM BEEBUG  APRIL 1985
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 REM C1=Number of Notes for
 110 REM Channel 1 etc
 120 C1=46:C2=30:C3=29
 130 :
 140 REM 1st Subscript Refers to:
 150 REM 1 - Channel Number/Attributes
 160 REM 2 - Envelope Number
 170 REM 3 - Pitch Value
 180 REM 4 - Duration
 190 :
 200 DIM Chan1(4,C1)
 210 DIM Chan2(4,C2)
 220 DIM Chan3(4,C3)
 230 :
 240 Scale$="  C  C# D  D# E  F  F# G
 G# A  A# B"
 250 Key=1
 260 Tempo=1
 270 :
 280 ENVELOPE1,1,0,0,0,0,0,0,126,-2,0,
-5,126,100
 290 ENVELOPE2,4,0,0,1,1,0,1,63,-1,0,-
10,126,100
 300 ENVELOPE3,1,0,0,0,0,0,0,126,-4,-1
,-4,126,100
 310 :
 320 REM Channel 1
 330 FOR N=1 TO C1
 340 PROCChan
 350 Chan1(1,N)=Chan
 360 IF Note$="R" Env=0 ELSE IF N=5 OR
 N=11 OR N=25 OR N=30 OR N=36 OR N=42 E
nv=2 ELSE Env=1
 370 Chan1(2,N)=Env
 380 PROCAnalyseNote
 390 Chan1(3,N)=Pitch
 400 Chan1(4,N)=Duration
 410 NEXT N
 420 PRINT"Channel 1 Complete"
 430 :
 440 REM Channel 2
 450 FOR N=1 TO C2
 460 PROCChan
 470 Chan2(1,N)=Chan
 480 IF Note$="R" Env=0 ELSE Env=3
 490 Chan2(2,N)=Env
 500 PROCAnalyseNote
 510 Chan2(3,N)=Pitch
```

```
  520 Chan2(4,N)=Duration
  530 NEXT N
  540 PRINT"Channel 2 Complete"
  550 :
  560 REM Channel 3
  570 FOR N=1 TO C3
  580 PROCChan
  590 Chan3(1,N)=Chan
  600 IF Note$="R" Env=0 ELSE Env=1
  610 Chan3(2,N)=Env
  620 PROCAnalyseNote
  630 Chan3(3,N)=Pitch
  640 Chan3(4,N)=Duration
  650 NEXT N
  660 PRINT"Channel 3 Complete"
  670 :
  680 Ch1=0:Ch2=0:Ch3=0
  690 :
  700 REPEAT
  710 IF ADVAL(-6)>0 AND Ch1<C1 Ch1=Ch1
+1:SOUNDChan1(1,Ch1)+1,Chan1(2,Ch1),Cha
n1(3,Ch1),Chan1(4,Ch1)*Tempo
  720 IF ADVAL(-7)>0 AND Ch2<C2 Ch2=Ch2
+1:SOUNDChan2(1,Ch2)+2,Chan2(2,Ch2),Cha
n2(3,Ch2),Chan2(4,Ch2)*Tempo
  730 IF ADVAL(-8)>0 AND Ch3<C3 Ch3=Ch3
+1:SOUNDChan3(1,Ch3)+3,Chan3(2,Ch3),Cha
n3(3,Ch3),Chan3(4,Ch3)*Tempo
  740 UNTIL Ch1=C1 AND Ch2=C2 AND Ch3=C3
  750 :
  760 END
  770 :
 1000 DEF PROCChan
 1010 READ Note$:IF LEFT$(Note$,1)="&"
Chan=EVAL(Note$):READ Note$,Duration EL
SE Chan=0:READ Duration
 1020 ENDPROC
 1030 :
 1040 DEF PROCAnalyseNote
 1050 IF Note$="R" Pitch=255:ENDPROC
 1060 IF LEN(Note$)<2 OR LEN(Note$)>3 T
HEN PRINT"ERROR IN DATA ";Note$:PRINT"N
ote Number ";N:STOP
 1070 IF LEN(Note$)=2 THEN NoteName$=LE
FT$(Note$,1) ELSE NoteName$=LEFT$(Note$
,2)
 1080 Octave=VAL(RIGHT$(Note$,1))
 1090 Pitch=Key+INSTR(Scale$,NoteName$)
/3*4+(Octave-1)*48
 1100 IF Pitch<0 OR Pitch>255 THEN PRIN
T"ERROR IN PITCH DATA ";Note$;" Pitch =
 ";Pitch:PRINT"Note Number ";N:STOP
 1110 ENDPROC
 1120 :
 1130 REM Channel 1
 1140 DATA &200,B2,2,A2,2,G#2,2,A2,2
 1150 DATA &200,C3,4,R,4,D3,2,C3,2,B2,2
,C3,2
 1160 DATA &200,E3,4,R,4,F3,2,E3,2,D#3,
2,E3,2
 1170 DATA &200,B3,2,A3,2,G#3,2,A3,2,B3
,2,A3,2,G#3,2,A3,2
 1180 DATA &200,C4,8,A3,4,C4,2,G3,1,A3,1
 1190 DATA &200,B3,4,A3,4,G3,4,A3,2,G3,
1,A3,1
 1200 DATA &200,B3,4,A3,4,G3,4,A3,2,G3,
1,A3,1
 1210 DATA &200,B3,4,A3,4,G3,4,F#3,4
 1220 DATA &200,E3,8
 1230 :
 1240 REM Channel 2
 1250 DATA &200,R,8
 1260 DATA &200,A1,4,C2,4,C2,4,C2,4
 1270 DATA &200,A1,4,C2,4,C2,4,C2,4
 1280 DATA &200,A1,4,C2,4,A1,4,C2,4
 1290 DATA &200,A1,4,C2,4,C2,4,C2,4
 1300 DATA &200,E1,4,B1,4,B1,4,B1,4
 1310 DATA &200,E1,4,B1,4,B1,4,B1,4
 1320 DATA &200,E1,4,B1,4,B0,4,B1,4
 1330 DATA &200,E1,8
 1340 :
 1350 REM Channel 3
 1360 DATA &200,R,8
 1370 DATA &200,R,4,E2,4,E2,4,E2,4
 1380 DATA &200,R,4,E2,4,E2,4,E2,4
 1390 DATA &200,R,4,E2,4,R,4,E2,4
 1400 DATA &200,R,4,E2,4,E2,4,E2,4
 1410 DATA &200,R,4,E2,4,E2,4,E2,4
 1420 DATA &200,R,4,E2,4,E2,4,E2,4
 1430 DATA &200,R,4,E2,4,R,8
 1440 DATA &200,R,8
```

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

BUG IN ASSEMBLER - R.J. Head

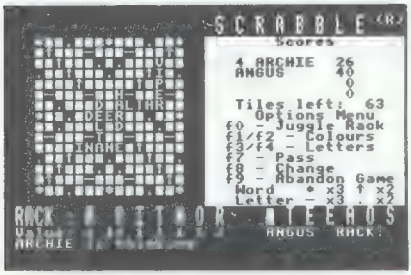
If you assign a zero page address to a variable, for use in a two pass assembler section of a program (eg. message=&80) you should assign it at the beginning of the assembler or in the Basic section of the program preceeding the assembler. This is because on the first pass the assembler assumes all addresses require a two byte address whereas a zero page address only requires one byte. Although the assembler 'realizes' that a single byte zero page address is required on the second pass, all branch instructions in the program will vector wrong because of the assembler's initial miscalculation.

# SCRABBLE FOR THE BEEB

## A new game from Leisure Genius

**Scrabble enthusiast Ian Tresman tries his skill against a new computer version of this popular game.**

```
Title    : SCRABBLE
Supplier : Leisure Genius
Price    : £12.95 (cassette)
           £15.95 (disc)
Rating   : ****
```




Love it or hate it, Scrabble ranks with chess, backgammon, and bridge, as one of the classic games of strategy.

Both disc and tape versions are now available for the BBC micro from Leisure Genius though there is no real difference between the two. Tape loading time is about 5 minutes during which time a mode 7 screen depicting the Scrabble board is displayed.

The option to play up to four hands is presented, any number of which can be played by the computer with a skill level from 1 to 4; in a two-handed game, this averages 160 and 320 points respectively.

Finally you are given the opportunity to look at the computer's rack during play, and see its best move while it is thinking. This is very enlightening, especially to the novice, who will be able to glean many useful strategic tips.

Response time from the computer is in the order of twenty to forty seconds and is well inside the British Tournament limit of two minutes. During a game this is not enforced.

A function key is provided to randomly juggle the letters in your rack. And there are also keys to allow you to either pass, or change some of the letters in your hand.

The majority of players will find that Scrabble plays a very competitive game. Its 8000 word dictionary is culled from the Scrabble players' bible: Chambers 20th Century Dictionary, New Edition. Interestingly, the choice of words has been deliberately chosen so as to give a 'fairer' game.

Computer Scrabble would play better if it had a more extensive two-letter word vocabulary. I discovered that the built-in dictionary only uses 40 of the possible 91 allowed two-lettered words. I would have liked to have seen the use on the higher skill levels of favourites such as: ai, ee, jo, ka, sh, yu, and zo.

The BBC machine is clearly at a memory disadvantage compared to the Apple and Spectrum versions of Scrabble whose 48K memory allowed 9100 and 11000 word vocabularies respectively, and, high-resolution displays. During the games I have played to date, I have been challenged as to the validity of some very basic words: largest, pigeon, menace, and moon. I was also curious about the computer making a plural of 'ozone', but this checked out alright.

Scrabble is a worthwhile long-awaited addition for the BBC microcomputer. It will not mind if you take thirty minutes a move, nor if you sneak and find from the dictionary that seven-letter word which you knew existed all along.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

OPENOUT BUG

Writing to an existing random access file that is write protected will generate an error but will also flush the buffer resulting in the first 256 characters of the output data in memory being lost.

Tested on O.S. 1.2
. Basic I & II
6502 2nd proc.

# CALCULATING THE LENGTH OF PROGRAMS

**The major drawback of Basic programming on the Beeb is the limited memory available. When memory space is at a premium, you need to know just how much is taken up by your program. Graham Crow describes a short utility which will calculate the length of all or part of any of your programs.**

This short utility will tell you how long your program is, in bytes of memory, so that you know exactly how much room you have left for data. The utility will also calculate the memory space taken up by a section of your program, even a single line. This is useful when calculating how much room you will save by removing a section of program, before you do it.

The utility is presented here as two functions (FNbytes and FNaddress), suitable to add onto the end of a program, along with a short demonstration program. This just prompts you for two line numbers and then prints the inclusive length of the program between these lines.

If one of the line numbers entered does not exist in the program, or the second number is smaller than the first, the program will halt with a short error message to tell you of the problem. You can measure the length of a single line by entering the same line number for each prompt.

When you use the functions in your own programs they are called, either in immediate mode or from within the program with

```
PRINT FNbytes(A,B)
```

or

```
L=FNbytes(A,B)
```

where A is the first line number and B the second. The function returns the length of program between them.

When you have typed in the functions and checked that they work with the demonstration program, delete the lines 10 to 150 and 10320 to 10370 and then save the function definitions themselves with:

```
*SPOOL (filename)
LIST
*SPOOL
```

Now they can be added to your own programs at any time with *EXEC (filename), as long as the highest line number in your own program is not greater than 10000.

PROGRAM NOTES

The short demonstration program between lines 90 and 140 simply prompts the user for two line numbers and uses these parameters for the call to the function, FNbytes, at line 120.

FNbytes itself calls a second function, FNaddress. This searches your program to find the address in memory of a program line. To understand the workings of both functions you need to know how Basic stores program lines in memory.

As well as the ASCII characters and Basic keyword tokens that make up a program line, the line number and the length of the line are also included. Each line is stored in the following form:

(&0D) (line no. high byte) (line no. low byte) (line length) (first character) ... (last character)

In addition, following the last line of the program there is:

(&0D) (&FF)

FNbytes finds the address of the start and finish lines, using FNaddress, and then adds together the line lengths of each line between them.

```
10 REM PROGRAM BYTE LENGTH
20 REM VERSION B0.1
30 REM AUTHOR  G.M.CROW
40 REM BEEBUG  APRIL 1985
50 REM PROGRAM SUBJECT TO COPYRIGHT
60 :
70 ON ERROR GOTO 10330
```

```
   80 :
   90 MODE 7
  100 INPUT TAB(8,10)"1st line No. ",L1
  110 INPUT TAB(8,12)"2nd line No. ",L2
  120 L=FNbytes(L1,L2)
  130 PRINT TAB(8,15)"length = ";L;" by
tes"
  140 END
  150 :
10000 DEF FNbytes(startline,endline)
10010 LOCAL address,startaddress,endadd
ress,bytes
10020 IF startline>endline THEN PRINT "
Error in lines":STOP
10030 REM Find address of start & end l
ines
10040 startaddress=FNaddress(startline)
:endaddress=FNaddress(endline)
10050 address=startaddress:bytes=0
10060 REM Use line length byte to count
10070 REPEAT
10080 bytes=bytes+address?3
10090 address=address+?(address+3)
10100 UNTIL address>endaddress
10110 REM Add 2 bytes if last line
10120 IF address?1=&FF THEN bytes=bytes
+2
10130 =bytes
10140 :
10150 DEF FNaddress(target)
10160 LOCAL low,high,none,address,add,l
ine
10170 low=PAGE:high=TOP:none=FALSE
10180 REM Use binary search method
10190 REPEAT
10200 address=INT(low+((high-low)/2))
10210 add=address
10220 REM Work back to start of line
10230 REPEAT add=add-1:UNTIL ?add=13
10240 REM but line length could be 13
10250 IF?(add-3)=13 THEN add=add-3
10260 line=(add?1)*256+add?2
10270 IF line>target THEN high=address
10280 IF line<target THEN low=address
10290 IF high-low<2 THEN none=TRUE
10300 UNTIL line=target OR none
10310 IF none THEN PRINT"No such line":
STOP ELSE =add
10320 :
10330 MODE 7
10340 ON ERROR OFF
10350 IF ERR=17 THEN END
10360 REPORT:PRINT " at line ";ERL
10370 END
```

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

SMALL OPENOUT FILES - A. McDonald

The operating system defaults to a length of 16K for random access files. Shorter files, less wasteful on disc space, can be created by saving a dummy file of the required length first and then accessing the file with OPENUP.

```
SAVE "name" 0000 +00FF
```

This creates a file only 256 bytes long.

PRINTING HEX NUMBERS - K. Kilmoore

To assign a string a number in hexadecimal notation the tilde (~) sign is used but in a different manner to that used when printing out the number. The following will print out the value of N in hexadecimal notation.

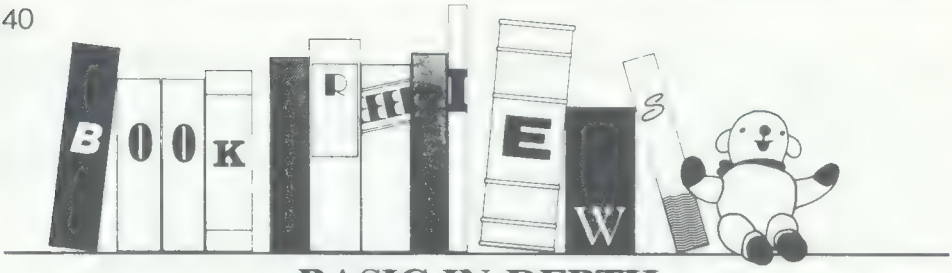
```
HEX$=STR$~(N)
PRINT HEX$
```

OC TROUBLES IN WORDWISE

The OC (output code) embedded command in Wordwise can cause problems if more than about 20 codes are output in one line, as for example when a lot of subscripting is needed in a mathematical expression. The other commands affecting paging and justification can be corrupted.

ZERO PAGE CORRUPTION - Bill Walker

Although the Basic ROM does not use zero page locations &70 to &8F, the same does not go for many other ROMs. View (1.4) for example uses &84 and &85 whenever it is called upon by the operating system, i.e. whenever an unrecognized *command is issued from within ANOTHER ROM in a machine containing the View ROM. This means that a program using these locations to store data that issues, say, a disc filing system command (such as *CAT) could (depending on the order of the ROMs in the machine) corrupt the data.

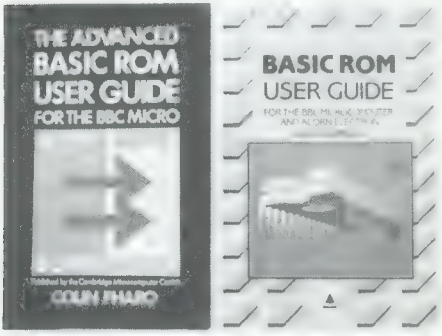



# BASIC IN DEPTH

**Following the highly successful Advanced User Guide to the BBC micro's operating system ROM, two books have appeared recently claiming to do the same for the Beeb's Basic Rom. Alan Dickinson has been dipping into these two books and now reports.**

Most of us will have wondered what makes the Beeb tick, (apart from the motor relay!), and perhaps puzzled over the hidden secrets of the Basic ROM chip. Brave souls who attempt machine code programming usually run into the problems of needing to handle floating point numbers, or perhaps require a random number generator, and would dearly love to know how Acorn managed it. Now there are two books available that help to answer all these questions.

Both books are specifically aimed at Basic I and Basic II users and are not suitable for HiBasic, USBasic (in Beebs for the USA), or any subsequent releases of BBC Basic. In addition both books really require some knowledge of machine code programming before they start to make much sense.




The Advanced Basic ROM User Guide by Colin Pharo. Published by Cambridge Microcomputer Centre at £8.95.

This is a spiral bound 182 page volume, in the style of the Advanced User Guide from the same publishers. It's a neat layout, but one which seems to incorporate liberal quantities of expensive white space.

The book contains a brief description of compilers and interpreters, and an explanation of the numbering systems used within BBC Basic, but is mainly concerned with describing 69 subroutines contained within the Basic ROM.

The routines are grouped according to the type of data that they handle, integer, floating point, conversions, trigonometry, and random number functions. Each section consists of some introductory text, a summary list of the routines, a detailed description of each routine, (one routine per page), and a simple demonstration example of using it. An approximate timing is given for each routine, which must be invaluable information for anyone seeking to optimise Basic programs. For example, I was amazed to find that the SIN routine typically takes 15,000 microseconds, whilst TAN consumes 41,000.

The book is completed by a Basic memory map, notes on timings, a small section on trigonometric methods, and a very informative section concerning linkage of large machine code programs.

The Basic ROM User Guide by Mark Plumbley. Published by Adder at £11.45.

This 359 page paperback is one of the first books from yet another Cambridge publishing company. It is

crammed with information without appearing cluttered, and a welcome addition to the library.

The early chapters of the book are concerned with the architecture of the BBC Basic system, including discussion on how Basic handles data types, tokenizes programs, and implements control structures such as procedures, loops, and ON controls. Assemblers and disassemblers, overlaying procedures, adding new commands to Basic, and error trapping are all covered in detail in the first 160 pages, whilst the remainder of the book contains details of some 80 ROM routines, a very full description of Basic error codes, and a set of tables covering Basic memory maps, token values, etc.

The book contains a wealth of programs, a disassembler, a partial renumber utility, and a 'bad program' recovery routine, though it must be said that none of these are dazzlingly original to BEEBUG readers. The book is an excellent compromise between being an informative readable guide and a useful reference book.

Neither of these books is for the beginner, and neither of them is suitable as an introduction to the mysteries of assembly language programming. Both are interesting and informative; Colin Pharo's book is particularly good for its explanation of trigonometry and program linkage, but on the whole I prefer the more expensive book by Mark Plumbley. This, I feel, represents excellent value for money, and is likely to be of more interest to all those who are not experts in machine code, but simply want to know more about the inner workings of BBC Basic.

The Advanced Basic ROM user Guide is available from BEEBUGSOFT to BEEBUG members for £7.95 all inclusive.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

DATA REMARKS - A. Roberts

Although REM statements can be added into DATA statements, a comma in the REM statement will delimit the REM and re-enable the DATA. For example in the following program the value assigned to the variable Z is 10 and not 3 as you'd expect:

```
10 READ X,Y,Z
20 DATA 1,2:REM a remark with a comma,10
30 DATA 3
```

RECOVERING LOST PROGRAMS WITH Z80 - Chang Sing Pang

Z80 owners who lose programs by switching over to CP/M without saving their work first can recover the program by creating a dummy file with:

```
SAVE 0 <filename>
```

and then load the dummy file with:

```
<filename>
```

The program can then be recovered simply with OLD.

Z80 BASIC STRING BUG - Chang Sing Pang

In Z80 Basic on the Z80 second processor (unlike 6502 Basic), strings held on disc are considered to end at the first Return. So the following program will only display the first half of the string.

```
10 A$="first"+CHR$13+CHR$10+"second"
20 C=OPENUP"BUG"
30 PRINT#C,A$
40 PTR#C=0
50 INPUT#C,B$
60 PRINT B$
70 CLOSE#C
```

BEGINNERS THIS WAY

# INTRODUCING MACHINE CODE (Part 3)

Tested on O.S. 1.2 Basic I & II

**This month Gordon Weston continues his series on machine code for beginners by looking at further ideas on loops, and describes how to store and retrieve simple lists of numbers.**

In the last article, I gave you a little exercise in which line 140 of Program 7 was to be changed to LDX #0. When you decrement X by one on the first pass through the printing loop, the value in X becomes 255 and does not become zero until another 255 passes through the loop have occurred. The end result is that 256 characters are printed on the screen.

I also said that the fastest method of loop counting was to count downwards rather than upwards. First of all, to count upwards you can increment (add one to) the X or Y registers using the instructions 'INX' or 'INY'. To detect the loop exit condition, we need to introduce a COMPARE instruction, in which a 'pretend' subtraction takes place between the comparison value and the value in the register, without affecting the value in that register. CMP #40 would compare the value in the Accumulator with 40 and the instructions CPX #40 and CPY #40 would compare the contents of the X and Y registers with 40. This is illustrated in the two examples below. In example 2, 40 is subtracted from the contents of the X register at line 130 without affecting its contents. If the result of this pretend subtraction is ZERO then the branch no longer occurs.

```
Example 1
100 LDX #40
110 .loop
120 DEX
130 BNE loop
```

```
Example 2
100 LDX #0
110 .loop
120 INX
130 CPX #40
140 BNE loop
```

As you can see, both examples loop forty times, but example 2 is one instruction longer. More importantly, because that extra instruction is in a loop that is used forty times, then the program is effectively forty instructions longer.

Now we are ready for some new ideas and we can make a start by entering program 5 from the last article (repeated below).

```
Program 5
  10 MODE7
  20 DIM code 100
  30 FOR I%=0 TO 3 STEP 3:P%=code
  40 [
  50 OPT I%

 500 ]
 510 NEXT
 520 CLS:CALL start
 530 END
```

Now the CMP instruction can be very useful if we want a loop that will continue until a particular condition occurs, rather like a REPEAT-UNTIL loop in Basic. Insert these assembly lines with the skeleton (Program 5) to produce Program 8 and run the program.

```
Part Program 8
  100 .start
  120 .loop
  130 JSR &FFE0
  140 CMP#13
  150 BNE round
  160 RTS
  170 .round
  220 JSR &FFEE
  250 JMP loop
```

This program will display on the screen all text typed in at the keyboard until terminated by a Return. Line 130 reads the next character from

the keyboard and the ASCII value of the key pressed is stored in the Accumulator for line 220 to print to the screen. The 'JMP loop' instruction at line 250 forces the program to restart at line 120 and the only way to escape from this loop is at line 140. Here, the contents of the Accumulator are compared with 13, which is the ASCII value for the Return key. If the contents of the Accumulator are NOT EQUAL to 13, then the program branches round line 160 to line 170. If the contents of the Accumulator EQUAL 13 then the program reaches 'RTS' at line 160 and returns to Basic.

Besides deciding if a number is equal to that stored in a register, you can also decide if a number is greater than, or smaller than that stored in a register. We use two new instructions called 'BCC' (Branch if Carry Clear) and 'BCS' (Branch if Carry Set) where the term 'CARRY' refers to one of the FLAGS in the 8 bit status register. Rather than going into detail at this stage, it is better to select which branch instruction you need from the following rules.

After CMP #data....(with Accumulator)
a BCC branch occurs if A < data
a BCS branch occurs if A > data
or if A = data

Suppose we want to restrict the input from the keyboard to numerical keys only. We have to know their ASCII values and if you refer to page 486 of the User Guide you will see that number keys are in the range ASCII code 48 to ASCII code 57. We want to branch back to the start if the Accumulator value is less than 48, and also branch back to the start if the Accumulator value is equal to, or greater than, 58.

You can enter these additional lines to Program 8 and run the program which then only allows numerical characters to be printed to screen.

```
180 CMP #48
190 BCC loop
200 CMP #58
210 BCS loop
```

Obviously, these lines would reject the value 13 which is why CMP #13 and RTS appear earlier in the program loop.

Since this is a simple input routine, the next thing to consider is how to store what is being entered. To make a better demonstration we will use part of the screen memory to store our input. The Mode 7 screen takes 1000 bytes from address &7C00 to &7FE7 and the address &7E58 refers to the position 16 lines down on the left hand side of the screen (don't worry how these addresses are calculated - they are just convenient for this example).

> Model A users with 16K of memory will need to change these addresses from &7C00 to &3C00, &7FE7 to &3FE7 and &7E58 to &3E58.

Add line 230 STA &7E58 (STore the contents of the Accumulator in address &7E58) and every time a number key is pressed, that number, besides being printed at the top of the screen in the normal way, will also overwrite the existing number printed half way down the screen as it is stored in the screen area of memory.

This storage instruction uses the same address in memory all the time. A more flexible storage instruction takes the form STA &7E58,X where the contents of the Accumulator are stored at the address &7E58+X. In other words, if X=0 the Accumulator is stored at address &7E58, but if X=1 then the Accumulator is stored at address &7E59 and so on. In assembler this is the way to store a series of values at consecutive addresses, just as in an array in Basic. STA &7E58,Y works in the same way.

Enter the new lines 110, 230 and 240 to form Program 9 below. Each time a valid character is stored at line 230 the X register is incremented in preparation for storing the next valid character.

Part program 9

```
100 .start
110 LDX #0
120 .loop
130 JSR &FFE0
140 CMP #13
150 BNE round
160 RTS
170 .round
```

```
180 CMP #48
190 BCC loop
200 CMP #58
210 BCS loop
220 JSR &FFEE
230 STA &7E58,X
240 INX
250 JMP loop
```

Try entering more than 256 numbers and you will find that although printing continues in the top half of the screen, in the bottom half of the screen the new numbers start overwriting existing numbers because the value in X has changed from 255 back to 0 and is now counting upwards again. Program 9 is only a simple demonstration model as it lacks the facility to limit the number of characters input, it does not allow editing of the stored input data and does not give any warning when mistakes are made, but it does show a useful technique in action. Note that storing values directly in the screen memory area is a fast technique often used in games and other machine code programs, but is not generally accepted as good programming practice (hence the need for changes by model A users).

The final part of this short series on machine code for beginners will appear in the next issue of BEEBUG.


# POSTBAG

BOXING CLEVER

I have enclosed a very simple Break key 'guard'. It consists simply of a piece of card cut to size (3" long by 1" high), folded and sellotaped to form an open box shape. This then sits over the Break key (or any other same size key) to guard against accidental operation. I find this most useful when playing games.

Philip Baum

We have tried out the example that Philip sent, and it actually is quite a useful idea for preventing any fatal finger fumbling.

LEAP IN THE DARK

I wonder if you are aware that there is a small error in the "Cartoon Calendar" program in BEEBUG Vol.3 No.7. This excellent program as listed does not correctly take into account the extra day in leap years. The fault is, of course, in line 1310 which should read:

```
1310 Day=Fdays%+3+Leap%
```

(note the '+' instead of '-').

D.Shaul

Our thanks to Mr Shaul for spotting this slip on our part. The cartoon competition that we ran with this program attracted a lot of interest and the results are in this month's supplement.


# POSTBAG

MUSIC MAESTRO PLEASE

I was surprised to read in the December issue of BEEBUG, a review by Steve Ibbs of a book I bought two months previously. The title is 'Making Music on the Beeb' by Ian Waugh. As a newcomer to computing a quick glance at the book had convinced me that it was way above my head. I was pleased to read what was actually in the book. Mr Ibbs made it sound quite interesting even if he did prefer the other book he reviewed.

I started to read the book with renewed interest, and have now tried out some of the programs and various sound effects, truly amazing! Thank you Mr Ibbs for opening my eyes.

Charles Harvey

KENNETH KENDAL'S HICCUPS

Some months ago I had fitted (professionally) the Acorn PHROM speech system, and at first all was well but alas my poor old Kenneth Kendal has got the hiccups. When switching on my machine from cold and using the speech system, all words are correctly reproduced except 'G' and 'GOOD' (words 201 and 202). These are produced with a hiccup at the end which mysteriously clears itself after the machine has been on for about 15 minutes. Please advise me if the PHROM is faulty or have I got to knit it a little woollen

jumper to keep it warm so poor Kenneth doesn't get the flu.

A.Bonser

Our own copy of Kenneth is fortunately alive and well. We cannot undertake much diagnosis at a distance and suggest you take your ailing patient to the nearest Beeb surgery (or dealer).

SOMETHING NEW ON THE MENU

Did you know that the new DFS (1.2), that is part of the second processor DNFS chip, will not allow your MENU program (BEEBUG Vol.2 No.4) to work correctly? Acorn, in their infinite wisdom decided to change the addresses where the information on the file is stored, and didn't actually tell anyone. The solution is simple - just change line 700 to read:

```
700 base=&70:old=&BC:length=&CO
```

Nick Clark

We have checked this out and Nick Clark's modification certainly does work. It is also encouraging to find that BEEBUG programs do have such a long life. We have decided that in future we shall also be checking BEEBUG programs for compatibility with the 6502 2nd processor as well as O.S. 1.2 and Basic I & II.

A CAUTIONARY TALE

I purchased a well known make of disc drive in July. It went wrong in November and was returned (a round trip of 40 miles). It was collected in December (another 40 miles) but did not work! The boss had been ill, and so the repair had not been checked.... The item was returned yet again and I am still waiting for it to be properly repaired. Moral - always ask if equipment has been checked both when you buy and when repaired, and by the retailer and not just by the factory.

Don Maskell

That sounds like quite a drive....!

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

CORRECT INTEGREX MODE 7 DUMPS - J.A.Allmond

The Integrex Colourjet printer can dump mode 7 screens correctly (despite the remarks in the review in BEEBUG Vol.3 No.8) if the American character set is selected with the DIP switches or by selecting this character set by software using VDU1,27,82,1 before the dump and reseting to the English character set with VDU1,27,114 afterwards.

MORE COMMANDS IN MEMOPLAN - A.A.N. Ewing

Owners of the Z80 second processor with Memoplan can access two extra menus not mentioned in the manual using CTRL-X and CTRL-X-X. These menus contain most of the commands available from the functions keys but obtainable with a single letter from the menu - useful if you lose the key strip!

CENTRING THE MODE 7 SCREEN - C. Walker

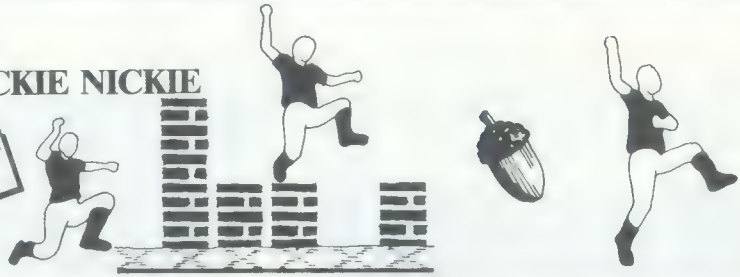
Centring text lines with an odd number of characters can be tricky in mode 7, especially if the left side of the screen is filled with control codes. Entering VDU23;0,2,52;0;0;0; shifts the whole display one character to the left. The effect is cancelled simply with any MODE command.

SIDEWISE RAM BENEFITS - Martin Parr

If you want some sideways RAM to play around with but cannot afford the very expensive 6264 chips needed for the 'Sidewise' board, you can use an inexpensive 6116 chip instead. This will only give you 2K of sideways RAM, but that may be enough for experimenting. You will need to carefully bend pins 24 and 21 on the chip out at right angles and solder about two inches of fine insulated wire to each. Now place the chip in the lower end of the RAM socket (i.e. chip pin 1 in socket position 3 and insert the wire from pin 24 into socket position 28 and that from pin 21 into position 27.

# BRICKIE NICKIE

Tested on O.S. 1.2
Basic I & II
6502 2nd proc



BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICK

**Master the lifts and moving platforms, avoid the falling bricks (or kick them out of the way) and escape from one hazardous world to the next in this most unusal game from Benedict Freeman.**

'Brickie Nickie' is a small guy trapped in a generator room. His life will soon end if he can't get to the exit quickly. Can you help him to survive?

Navigate the lifts and walkways and get past the brick walls, picking up acorns as you go, to gain the highest possible score. You will also have to avoid the bricks which fall from above, loosely aimed at your current position. The quicker you finish a sheet, the higher the score, but kicking the bricks away loses you points.

Your man is able to climb up and down one level at a time, and if he is confronted by a brick wall, then kicking the wall will dislodge the brick on or below his own level. As you will find out when you play the game, you will need some of the bricks as stepping stones to get from one catwalk to another - so don't kick them all away.

You start from the 'S' box and must reach the end box marked 'E' before the generator overheats and explodes (imminent explosion is indicated by a beeping sound). If you complete this task then you go on to the next screen, and so on until you lose three lives.

If the generator explodes, you will lose a life, or if a brick falls on your head (as it will do on numerous occasions when you first start to play) you will also have to apply for resurrection.

The keys to use are: 'Z' and 'X' for left and right, '*' and '?' for up and down and 'Shift' and ']' for kick right and kick left. The ']' key is situated between the '*' and 'Return' keys.

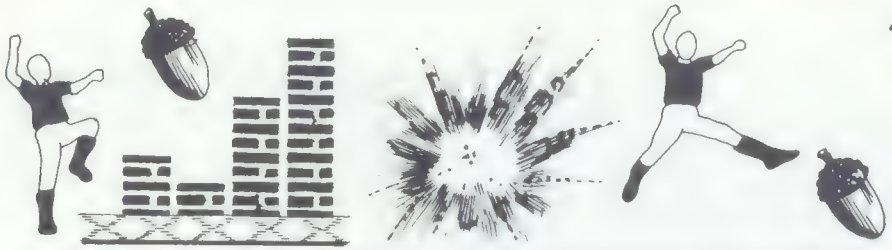
These keys are given when you run the program.

If you are running this program on a disc system then PAGE must first be set to &1200, but remember not to press Break as this corrupts the program.

PROGRAM NOTES

Because of the calculated RESTORE at line 1020, you will notice that there is a gap at the end of the program between lines 2620 and 5000. The data at the end of the listing MUST start at line 5000 and MUST follow the line numbering specifically.

The two keys to kick left and right are defined in lines 190 and 200, and the other keys are defined in the data statement at line 1420. The method used here is to read a negative inkey value from the data statement (-98 is for 'Z') and if the relevant key is being pressed then add the next two values from the data list to the X and Y co-ordinates of your man.



## BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICKIE BRICKIE NICK

FUNCTION AND PROCEDURE LIST

| | |
|---|---|
| screen/ screendetails/ print | Reads the variables and game layout and display layout on screen. |
| pos | Tests for brick to kick. |
| newpos | Checks for key pressed and updates X and Y. |
| move | Moves the man on the screen. |
| block | Positions and draws random bricks on screen. |
| dish | Move and draw platform. |
| lift | Move and draw lift. |
| init | Prints the instructions. |
| init2 | Sets up the variables. |
| check | Check for next screen. |
| end | Check for end of game. |
| chars | Sets up the user-defined characters. |
| kick | Kick the brick. |

```
   10 REM PROGRAM BRICKIE
   20 REM VERSION B0.3
   30 REM AUTHOR  B.FREEMAN
   40 REM BEEBUG  APRIL 1985
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
  100 ON ERROR GOTO 2580
  110 HS%=1000:PROCchars
  120 REPEAT:PROCinit
  130 MODE2:VDU23,1,0;0;0;0;
  140 REPEAT:CLS:CLG
  150 BX%=0:BY%=0:NBY%=0:NBX%=0
  160 PROCinit2:PROCscreen
  170 PROCscreendetails:TIME=0
  180 REPEAT:PROCmove
  190 IFINKEY-1 kright=FALSE:PROCkick
  200 IFINKEY-89 kright=TRUE:PROCkick
  210 PROClift:PROCdish:PROCblock
  220 screen=FALSE:PROCcheck
  230 IF TIME>11000 SOUND1,-15,1500,1
  240 IF TIME>13000 dead=TRUE
  250 VDU17,3:PRINTTAB(2,1);lives;TAB(8
1);S%;CHR$32;TAB(17,1);level
  260 UNTIL dead OR screen
  270 IFscreen S%=S%+10000-TIME:FORI=10
0TO200STEP10:SOUND1,-15,I,3:NEXT:S%=S%+
(500*(level-1))
  280 IF NOT screen PROCend
  290 UNTIL new:UNTILFALSE
  300 END
  310 :
 1000 DEFPROCscreen
 1010 IFlevel=5THENlevel=1
 1020 RESTORE(5000+(level*80)-80)
 1030 READX%,Y%,ENX%,ENY%,EXX%,EXY%,FS%
,FX,FY,FT,L%,D%,LX%,LY%,lmax,lmin,DX%,D
Y%,dmax,dmin,GX%,GY%
 1040 NX%=X%:NY%=Y%
 1050 REPEAT
 1060 READA%,B%,C%,T%:PROCprint
 1070 UNTILC%=0
 1080 ENDPROC
 1090 :
 1100 DEFPROCscreendetails
 1110 VDU17,1,31,X%,Y%,man2%,11,8,man%,
17,2,31,ENX%,ENY%,EN3,EN4,8,8,11,EN1,EN
2,17,1,31,EXX%,EXY%,EX3,EX4,8,8,11,EX1,
EX2,17,2,31,GX%,GY%,G3,G4,8,8,11,G1,G2
 1120 VDU 17,4
 1130 IFL%VDU31,LX%,LY%,L,L1
 1140 IFD%VDU31,DX%,DY%,L,L1
 1150 VDU17,11,31,FX,FY,FT3,FT4,8,8,11.
FT1,FT2
 1160 MOVE0,950:DRAW1280,950
 1170 VDU 17,6:PRINTTAB(0,0)"LIVES   SC
ORE  LEVEL"
 1180 ENDPROC
 1190 :
 1200 DEFPROCprint
 1210 VDU31,A%,B%
 1220 IFT%=0VDU17,0
 1230 IFT%=1ORT%=2VDU17,5
 1240 IFT%>2VDU17,7
 1250 IFT%=3THENT%=2
 1260 IFT%<4THENPRINTSTRING$(C%,CHR$(25
5-T%))
 1270 IFT%=4PRINTSTRING$(C%,CHR$girdbl+
CHR$girdb2)
 1280 IFT%=7ORT%=8THENVDUdisc,disc1
 1290 IFT%=10VDU17,6:PRINTSTRING$(C%,CH
R$girdbl+CHR$girdb2)
 1300 IFT%=9THENVDU17,6:PRINTSTRING$(C%
,CHR$B)
 1310 ENDPROC
 1320 :
 1330 DEFFNpos(X,Y)
 1340 =POINT(X*64+24,(31-Y)*32+28)
 1350 :
 1360 DEFPROCnewpos
 1370 RESTORE1420
```

▶

```
 1380 FORI=1TO4:READAB%,CX%,CY%
 1390 IFINKEY(AB%)NX%=NX%+CX%:NY%=NY%+C
Y%
 1400 NEXT
 1410 IFX%=NX%ANDNY%=Y%-1THENNY%=NY%+1
 1420 DATA-98,-1,0,-67,1,0,-105,0,1,-73
,0,-1
 1430 ENDPROC
 1440 :
 1450 DEFPROCmove
 1460 NX%=X%:NY%=Y%:PROCnewpos
 1470 IFNX%=X%ANDNY%=Y%THENENDPROC
 1480 IFNX%<0ORNX%>19ENDPROC
 1490 A%=FNpos(NX%,NY%):B%=FNpos(NX%,NY
%-1):IF(A%<>0)OR(B%<>0ANDB%<>3)ENDPROC
 1500 IFFNpos(NX%,NY%+1)=0THENENDPROC
 1510 VDU31,X%,Y%,17,0,man%,11,8,man2%,
31,NX%,NY%,17,1,man2%,11,8,man%
 1520 X%=NX%:Y%=NY%:SOUND1,-11,3,1
 1530 ENDPROC
 1540 :
 1550 DEFPROCblock
 1560 IFNOTblock BX%=X%+RND(5)-2:BY%=3:
block=TRUE:IFBX%<=30ORBX%>=18BX%=3:VDU31
,BX%,BY%,17,6,B:ENDPROC
 1570 NBY%=BY%+1
 1580 IFX%=BX%ANDNBY%=Y%-1THENdead=TRUE
:ENDPROC
 1590 IF((BX%=DX%ORBX%=DX%+1)ANDNBY%=DY
%)OR((BX%=LX%ORBX%=LX%+1))ANDNBY%=LY%TH
ENblock=FALSE:VDU31,BX%,BY%,17,0,B
 1600 IFFNpos(BX%,NBY%)<>0block=FALSE:E
NDPROCELSEVDU31,BX%,BY%,17,0,B
 1610 BY%=NBY%
 1620 VDU31,BX%,BY%,17,6,B
 1630 ENDPROC
 1640 :
 1650 DEFPROCdish
 1660 IFdrightTHENNDX%=DX%+1ELSENDX%=DX
%-1
 1670 IFNDX%=dmaxORNDX%=dmin dright=NOT
dright:ENDPROC
 1680 IF(X%=DX%ORX%=DX%+1)ANDY%=DY%-1TH
ENdman=TRUEELSEdman=FALSE
 1690 VDU31,DX%,DY%,17,0,L,L1,31,NDX%,D
Y%,17,4,L,L1:DX%=NDX%
 1700 IFdmanVDU17,0,31,X%,Y%,man2%,11,8
,man%:ELSEENDPROC
 1710 IFdrightTHENX%=X%+1ELSEX%=X%-1
 1720 VDU31,X%,Y%,17,1,man2%,11,8,man%
 1730 ENDPROC
 1740 :
 1750 DEFPROClift
 1760 IFlup NLY%=LY%-1ELSENLY%=LY%+1
 1770 IFNLY%=1minORNLY%=1max lup=NOTlup
:ENDPROC
 1780 IF(X%=LX%ORX%=LX%+1)ANDY%=LY%-1TH
ENlman=TRUE:VDU17,0,31,X%,Y%,man%,11,8,
man2%:ELSElman=FALSE
 1790 VDU31,LX%,LY%,17,0,L,L1
 1800 LY%=NLY%
 1810 IFlupTHENIFlman Y%=Y%-1ELSEIFlman
THENY%=Y%+1
 1820 VDU31,LX%,LY%,17,4,L,L1
 1830 IFlmanVDU31,X%,Y%,17,1,man2%,11,8
,man%
 1840 ENDPROC
 1850 :
 1860 DEFPROCinit2
 1870 screen=FALSE
 1880 ENVELOPE1,1,5,5,-10,30,30,30,50,0
,0,1,100,100
 1890 newlevel=level+1
 1900 lup=TRUE:dright=TRUE
 1910 man%=241:man2%=242
 1920 EN1=227:EN2=228:EN3=229:EN4=230
 1930 EX1=231:EX2=232:EX3=233:EX4=234
 1940 G1=243:G2=244:G3=245:G4=246
 1950 L=235:L1=236
 1960 FT1=237:FT2=238:FT3=239:FT4=240
 1970 lman=FALSE:dman=FALSE
 1980 block=FALSE
 1990 B=253
 2000 dead=FALSE
 2010 girdbl=225:girdb2=226
 2020 fruit=FALSE
 2030 ENDPROC
 2040 :
 2050 DEFPROCinit:IF S%>HS% HS%=S%
 2060 VDU22,7:PRINTTAB(0,1)CHR$129"Scor
e:";CHR$130;S%;TAB(20)CHR$129"Hi-Score:
";CHR$130;HS%
 2070 FORA=5TO6:PRINTTAB(6,A)CHR$131CHR
$157CHR$129CHR$141"Brickie Nickie   "CH
R$156:NEXT
 2080 PRINTTAB(9,8)CHR$130"by  B.Freema
n."'''CHR$134TAB(5)"Keys to use:"
 2090 PRINT''CHR$133TAB(7)"Z - Left   X
 - Right"'CHR$133TAB(7)"* - Up     ? -
Down"'CHR$133TAB(3)"SHIFT - Kick   ] -
Kick"'CHR$133TAB(11)"Left       Right"
 2100 PRINT'''TAB(4)CHR$134"Press Any K
ey to Start.";
 2110 G=GET
 2120 lives=3:new=FALSE:S%=0:level=1
 2130 ENDPROC
 2140 :
 2150 DEFPROCcheck
 2160 IFNOTfruitAND(X%=FXORX%=FX+1)ANDY
%=FYTHENS%=S%+1000:VDU17,0,31,FX,FY,FT3
,FT4,8,8,11.FT1,FT2,17,1,31,X%,Y%,man2%
,11,8,man%:fruit=TRUE:SOUND1,1,10,20
 2170 IF(X%=EXX%-1ORX%=EXX%+1)ANDY%=EXY
%THENlevel=level+1:screen=TRUE
 2180 ENDPROC
 2190 :
 2200 DEFPROCend
 2210 *FX15,1
 2220 lives=lives-1
 2230 PRINTTAB(2,1);lives
```

```
 2240 SOUND 0,-15,5,10:IFlives=0 N$=" G
AME OVER ":FORI3=0TO2000:NEXT:FORI=1TOL
EN(N$):PRINTTAB(4+I,14)MID$(N$,I,1):FOR
I2=1TO200:NEXT:SOUND1,-15,I*20,1:NEXT:n
ew=TRUE:*FX21
 2250 IFlives=0 I=INKEY(200)
 2260 ENDPROC
 2270 :
 2280 DEFPROCchars
 2290 VDU23,254,255,66,36,24,24,36,66,2
55:VDU23,225,255,2,4,8,16,32,64,255
 2300 VDU23,226,255,64,32,16,8,4,2,255:
VDU23,227,127,128,128,135,132,132,132,1
35:VDU23,228,254,1,1,241,1,1,1,241:VDU2
3,255,0,0,0,0,0,0,0,0
 2310 VDU23,229,128,128,128,135,128,128
,128,127:VDU23,230,17,17,17,241,1,1,1,2
54:VDU23,231,127,128,128,135,132,132,13
2,135:VDU23,232,254,1,1,225,1,1,1,129
 2320 VDU23,233,132,132,132,135,128,128
,128,127:VDU23,234,1,1,1,225,1,1,1,254
 2330 VDU23,235,63,127,255,255,255,255,
127,63:VDU23,236,252,254,255,255,255,25
5,254,252:VDU23,237,0,0,0,1,3,3,0,6:VDU
23,238,0,0,128,192,224,224,128,32
 2340 VDU23,239,7,3,1,1,1,2,4,8:VDU23,2
40,96,192,128,0,0,0,0,0:VDU23,241,60,66
,66,66,60,24,189,189:VDU23,242,189,189,
60,60,20,20,20,54
 2350 VDU23,253,254,254,0,246,246,246,0
,254:VDU23,243,16,3,34,6,4,68,4,63:VDU2
3,244,16,192,72,96,32,36,32,255:VDU23,2
45,32,35,36,36,36,36,35,63
 2360 VDU 23,246,1,225,1,1,225,33,225,2
55
 2370 ENDPROC
 2380 :
 2390 DEFPROCkick
 2400 IFkright KX=X%+1ELSE KX=X%-1
 2410 KY=Y%+1
 2420 OX=KX:OY=KY-1
 2430 BT=FNpos(OX,OY-1)
 2440 found=FALSE
 2450 REPEAT
 2460 IFFNpos(KX,KY)=0found=TRUE
 2470 KY=KY-1
 2480 UNTILfound
 2490 KY=KY+2
 2500 IFFNpos(KX,KY)<>6ENDPROC
 2510 IF FNpos(OX,OY)<>6 AND FNpos(OX,O
Y+1)<>6 ENDPROC ELSE VDU 31,OX,OY,32
 2520 FORI=1 TO 100:NEXT
 2530 VDU31,KX,KY,32
 2540 IF BT=6 VDU31,OX,OY,17,6,B
 2550 IF S%>10 S%=S%-10
 2560 ENDPROC
 2570 :
 2580 ON ERROR OFF
 2590 MODE7:IF ERR=17 END
```


```
 2600 REPORT:PRINT" at line ";ERL
 2610 END
 2620 :
 5000 DATA17,28,18,28,18,7,1,18,15,1,-1
,-1,0,30,7,30,8,8,15,7,4,11
 5010 DATA2,30,18,1,2,16,5,4,17,16,3,1
 5020 DATA2,8,6,1,16,8,4,2,10,14,7,1,16
,15,2,1
 5070 DATA0,0,0,0
 5080 DATA4,6,2,6,17,24,1,18,13,1,-1,-1
,0,29,30,13,9,14,16,8,5,21
 5090 DATA0,31,9,4,2,14,7,1,18,14,2,1,2
,29,1,10
 5100 DATA15,29,4,2,15,28,4,2,15,27,4,2
,15,26,4,2
 5110 DATA15,29,4,2,15,28,4,2,15,27,4,2
,15,26,4,2,14,25,6,3
 5120 DATA7,11,3,1,2,7,2,4,6,8,2,1,5,12
,2,9,4,13,2,1
 5130 DATA17,14,1,9,17,15,1,9
 5140 DATA9,29,4,1,12,27,1,4,2,4,6,1
 5150 DATA0,0,0,0
 5160 DATA12,6,15,6,18,22,1,0,6,1,-1,-1
,6,21,23,12,3,7,10,1,8,9
 5170 DATA12,4,3,4,11,7,4,10,0,7,2,1,12
,8,2,9,14,8,2,9
 5180 DATA0,4,2,1,12,9,2,9,14,9,2,9,12,
10,2,9,14,10,2,9
 5190 DATA12,11,4,9,12,12,4,9,8,13,10,1
,4,22,2,1,3,24,1,10
 5200 DATA2,25,12,1,12,24,6,1,14,23,2,1
0,15,7,2,9,12,7,2,9,15,8,1,9
 5210 DATA15,9,1,9,15,10,1,9
 5230 DATA0,0,0,0
 5240 DATA3,12,1,12,18,7,1,2,26,1,-1,-1
,14,10,7,26,5,13,4,13,3,7
 5250 DATA1,10,4,1,1,13,4,1,18,5,2,1,16
,8,2,4,16,25,4,1,12,25,1,4
 5260 DATA10,23,1,4,8,24,2,1,6,25,2,1,1
,27,5,1
 5320 DATA0,0,0,0
```

IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues are kept in print (from April 1982) priced as follows:

Individual copies:
Volume 1 - £0.80
Volume 2 - £0.90
Volume 3 - £1.00
Volume 1 set (10 issues) £7
Volume 2 set (10 issues) £8

Please add cost of post and packing as shown:

| No of copies | DESTINATION UK | Europe | Elsewhere |
|---|---|---|---|
| 1 | 0.30 | 0.70 | 1.50 |
| 2 - 5 | 0.50 | 1.50 | 4.70 |
| 6 - 10 | 1.00 | 3.00 | 5.50 |
| 11 - 20 | 1.50 | 4.00 | 7.00 |

All overseas items are sent airmail (please send a sterling cheque). We will accept official UK orders but please note that there will be a £1 handling charge for orders under £10 that require an invoice. Note that there is no VAT on magazines.

This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the BEEBUG Reference Card and BEEBUG supplements are not supplied with back issues.

SUBSCRIPTIONS

Send all applications for membership, subscription renewals, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

U.K.
£6.40 for 6 months (5 issues)
£11.90 for 1 year (10 issues)

Eire and Europe
Membership £18 for 1 year.
Middle East £21
Americas and Africa £23
Elsewhere £25
Payment in Sterling essential.

PROGRAMS AND ARTICLES

All programs and articles used are paid for at up to £40 per page, but please give us warning of anything substantial that you intend to write. In the case of material longer than a page, we would prefer this to be submitted on cassette or disc in machine readable form using "Wordwise", "View", "Minitext Editor" or other means. If you use cassette, please include a backup copy at 300 baud.

HINTS

There are prizes of £5 and £10 for the best hints each month, plus one of £15 for a hint or tip deemed to be exceptionally good.

Please send all editorial material to the editorial address below. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

BEEBUG
PO Box 50
St Albans
Herts

Subscriptions & Software Address

BEEBUG
PO BOX 109
High Wycombe
Bucks HP10 8HQ

Hotline for queries and software orders

St.Albans (0727) 60263
Manned Mon-Fri 9am-4.30pm

24hr Answerphone Service for Access orders, and subscriptions

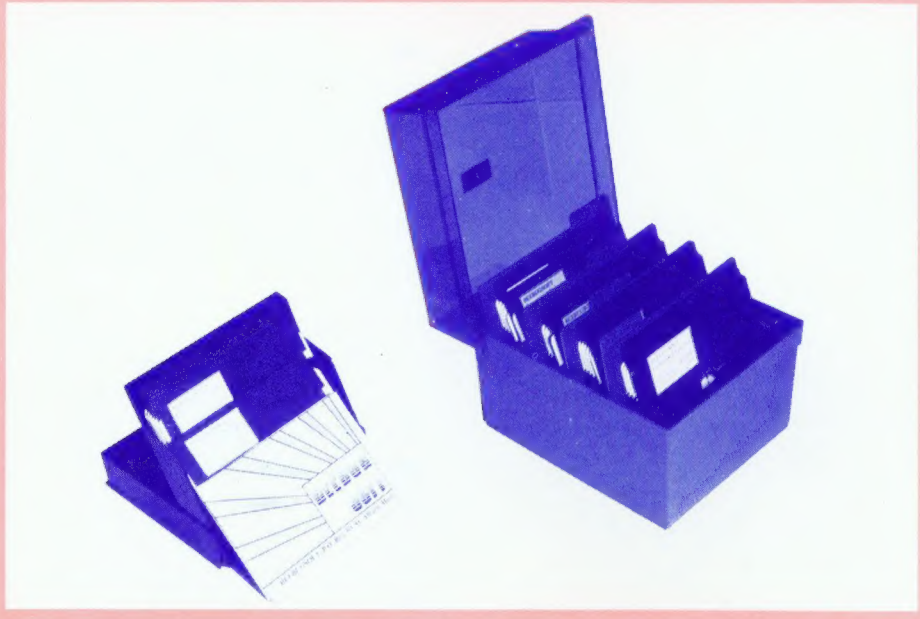
Penn (049481) 6666

If you require members' discount on software it is essential to quote your membership number and claim the discount when ordering.

BEEBUG MAGAZINE is produced by BEEBUG Publications Ltd.
Editor: Mike Williams.
Assistant Editor: Geoff Bains. Production Editor: Phyllida Vanstone.
Technical Assistant: Alan Webster.
Secretary: Debbie Sinfield
Managing Editor: Lee Calcraft.
Thanks are due to Sheridan Williams, Adrian Calcraft, John Yale, and Tim Powys-Lybbe for assistance with this issue.

# High Quality Low Priced Discs

## Backed by The Reputation of BEEBUG



| | |
|---|---|
| 10 S/S D/D Discs – £13.90 | 10 D/S D/D Discs – £19.40 |
| 25 S/S D/D Discs – £33.45 | 25 D/S D/D Discs – £46.95 |
| 50 S/S D/D Discs – £59.30 | 50 D/S D/D Discs – £87.05 |

All Prices Include Storage Box, VAT and Delivery to Your Home (UK).

All discs are 100% individually tested, supplied with hub ring as standard, and guaranteed error free. They are ideal for use on the BBC Micro and have performed perfectly in extensive tests at BEEBUG over many months.

Orders for 25 or 50 are delivered in strong plastic storage boxes with four dividers. Orders for 10 are sent in smaller hinged plastic library cases.

We are also able to offer the empty storage container, which holds up to 50 discs for £10 including VAT and post.

Please use the order form enclosed
or order directly from:
BEEBUGSOFT, P.O. Box 109,
High Wycombe, Bucks HP10 8HQ.

BEEBUG SOFT

# THE BEEBUG MAGAZINE ON DISC AND CASSETTE

The programs featured each month in the BEEBUG magazine are now available to members on disc and cassette.

Each month we will produce a disc and cassette containing all of the programs included in that month's issue of BEEBUG. Both the disc and the cassette will display a full menu allowing the selection of individual programs and the disc will incorporate a special program allowing it to be read by both 40 and 80 track disc drives. Details of the programs included in this month's magazine cassette and disc are given below.

Magazine cassettes are priced at £3.00 and discs at £4.75.
SEE BELOW FOR FULL ORDERING INFORMATION.

This Month's Programs Include:

Mixing Modes, a utility and demonstration program showing how to mix modes on the same screen; the complete BEEBUG Spreadsheet Program (combines parts 1 and 2); a program for playing multi-part music from our series 'Making Music on the Beeb'; Backwards, a short fun program for our April issue; machine code examples from our 'Beginners' series; a utility for calculating the length of all or part of a program; two sorting routines and demonstration program from this month's BEEBUG Workshop; Brickie Nickie, a highly attractive multi-screen game; and two extra items, a clever artificial intelligence program to keep you all guessing and a super machine code arcade game called Cosmonaut.

## MAGAZINE DISC/CASSETTE SUBSCRIPTION

Subscription to the magazine cassette and disc is also available to members and offers the added advantage of regularly receiving the programs at the same time as the magazine, but under separate cover.

Subcription is offered either for a period of 6 months (5 issues) or 1 year (10 issues) and may be backdated if required. (The first magazine cassette available is Vol. 1 No. 10; the first disc available is Vol. 3 No. 1.)

## MAGAZINE CASSETTE SUBSCRIPTION RATES

| | | |
|---|---|---|
| 6 MONTHS | (5 issues) | UK £17.00 INC... Overseas £20.00 (No VAT payable) |
| 1 YEAR | (10 issues) | UK £33.00 INC... Overseas £39.00 (No VAT payable) |

## MAGAZINE DISC SUBSCRIPTION RATES

| | | |
|---|---|---|
| 6 MONTHS | (5 discs) | UK £25.50 INC... Overseas £30.00 (No VAT payable) |
| 1 YEAR | (10 discs) | UK £50.00 INC... Overseas £56.00 (No VAT payable) |

## CASSETTE TO DISC SUBSCRIPTION TRANSFER

If you are currently subscribing to the BEEBUG magazine cassette and would prefer to receive the remainder of your subscription on disc, it is possible to transfer the subscription. Because of the difference between the cassette and disc prices, there will be an extra £1.70 to pay for each remaining issue of the subscription. Please calculate the amount due and enclose with your order.

## ORDERING INFORMATION

Please send your order to the address below and include a sterling cheque. Postage is included in subscription rates but please add 50p for the first item and 30p for each subsequent item when ordering individual discs or cassettes in the UK. Overseas orders please send the same amount to include the extra post but not VAT.
SEND TO:

**BEEBUGSOFT, PO BOX 109, HIGH WYCOMBE, BUCKS, HP10 8HQ**

Printed in England by Staples Printers St Albans Limited at The Priory Press.

ISSN 0263-7561